2001-03

Development, simulation and evaluation of the IEEE 802.11a physical layer in a multipath environmer

Tan, Kok Chye.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DEVELOPMENT, SIMULATION AND EVALUTION OF THE IEEE 802.11a PHYSICAL LAYER IN A MULTIPATH ENVIRONMENT

by

Kok Chye Tan

March 2001

Thesis Advisor: John McEachen
Second Reader: Xiaoping Yun

Approved for public release; distribution is unlimited

DEVELOPMENT, SIMULATION AND EVALUTION OF THE IEEE 802.11a PHYSICAL LAYER IN A MULTIPATH ENVIRONMENT

Kok Chye Tan-Civilian, Defence Science And Techology Agency (Singapore)
B. Eng., University Of Glasgow, 1993
Master of Science in Electrical And Computer Engineering–March 2001
Advisor: John McEachen, Department of Electrical And Computer Engineering
Second Reader: Xiaoping Yun, Department of Electrical And Computer Engineering

This thesis describes the development of a simulation of the newly proposed IEEE 802.11a physical layer and demonstrates the effects of Additive White Gaussian Noise (AWGN) and multipath on its performances. The IEEE 802.11a standardization group has selected Orthogonal Frequency Division Multiplexing (OFDM) as the basis for the new 5 GHz standard, targeting a range of data rates from 6 up to 54 Mbps.

Coded OFDM (COFDM) is a channel coding and modulation scheme which mitigates the adverse effects of fading by using wideband multicarrier modulation combined with time interleaving and a convolutional error correcting code. A guard interval is inserted at the transition between successive symbols to absorb the intersymbol interference created by the time domain spread of the mobile radio channel. The decoding process is performed using differential demodulation in conjunction with a hard decision Viterbi decoder.

The simulation results showed that COFDM system is capable of indoor environment communications in the presence of known multipath and noise conditions. The results obtained also showed that the COFDM configuration is immune to Doppler shift of 5 to 15 Hz.

.

DoD KEY TECHNOLOGY AREA: Command, Control and Communications.

KEYWORDS: Coded Orthogonal Frequency Division Multiplexing (COFDM), MATLAB, Convolutional Encoding, Viterbi Decoder, Interleaver, Multipath, Additive White Gaussian Noise (AWGN), Inverse Fast Fourier Transform (IFFT), Guard Interval

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2001	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Development, Simulation and Evalution Of The IEEE 802.11a Physical Layer In A Multipath Environment			**5. FUNDING NUMBERS**
6. AUTHOR(S) Kok Chye Tan			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department Of Defense 9800 Savage Rd Ft. Meade, MD 20755 ATTN : R531			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

This thesis describes the development and simulation of the newly proposed IEEE 802.11a physical layer and demonstrates the effects of Additive White Gaussian Noise (AWGN) and multipath on its performances. The IEEE 802.11a standardization group has selected Orthogonal Frequency Division Multiplexing (OFDM) as the basis for the new 5 GHz standard, targeting a range of data rates from 6 up to 54 Mbps. Coded OFDM (COFDM) is a channel coding and modulation scheme which mitigates the adverse effects of fading by using wideband multicarrier modulation combined with time interleaving and a convolutional error correcting code. A guard interval is inserted at the transition between successive symbols to absorb the intersymbol interference created by the time domain spread of the mobile radio channel. The decoding process is performed using differential demodulation in conjunction with a hard decision Viterbi decoder. The simulation results shown a COFDM system capable of indoor environment communications in the presence of known multipath and noise conditions. The results obtained also show that the COFDM configuration is immune to Doppler shift of 5 to 15 Hz.

14. SUBJECT TERMS Coded Orthogonal Frequency Division Multiplexing (COFDM), MATLAB, Convolutional Encoding, Viterbi Decoder, Interleaver, Multipath, Additive White Gaussian Noise (AWGN), Inverse Fast Fourier Transform (IFFT), Guard Interval			**15. NUMBER OF PAGES**
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DEVELOPMENT, SIMULATION AND EVALUTION OF THE IEEE 802.11a PHYSICAL LAYER IN A MULTIPATH ENVIRONMENT

Kok Chye Tan
Civilian, Defence Science And Technology Agency (Singapore)
B. Eng., University Of Glasgow, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2001

Author: _____
Kok Chye Tan

Approved by: _____
John McEachen, Thesis Advisor

_____
Xiaoping Yun, Second Reader

_____
Jeffrey B. Knorr, Chairman
Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis describes the development and simulation of the newly proposed IEEE 802.11a Wireless Local Area Network (WLAN) physical layer and demonstrates the effects of Additive White Gaussian Noise (AWGN) and multipath on its performances. The IEEE 802.11a WLAN standardization group has selected Orthogonal Frequency Division Multiplexing (OFDM) as the basis for the new 5 GHz standard, targeting a range of data rates from 6 up to 54 Mbps. Coded OFDM (COFDM) is a channel coding and modulation scheme which mitigates the adverse effects of fading by using wideband multicarrier modulation combined with time interleaving and a convolutional error correcting code. A guard interval is inserted at the transition between successive symbols to absorb the intersymbol interference created by the time domain spread of the mobile radio channel. The decoding process is performed using differential demodulation in conjunction with a hard decision Viterbi decoder. The simulation results show a COFDM system is capable of indoor environment communications in the presence of known multipath and noise conditions. The results obtained also show that the COFDM configuration is immune to Doppler shift of 5 to 15 Hz.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I would like to thank my employer, Defence Science And Technology Agency (Singapore) for giving me the opportunity to pursue my Msc study in Naval Postgraduate School. I thank Prof McEachen, my thesis advisor, for his encouragement and strong support given to me. I also thank Prof Yun, my second reader, who offered valuable advices along the way. Last but not least, special thanks to my wife, Lin Mei and my one month old baby boy, Yang Yi, for their love, understanding and encouragement.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This thesis describes the simulation of the newly proposed IEEE 802.11a physical layer and demonstrates the effects of Additive White Gaussian Noise (AWGN) and multipath on its performance. The IEEE 802.11a standardization group has selected Orthogonal Frequency Division Multiplexing (OFDM) as the basis for the new 5 GHz standard, targeting a range of data rates from 6 up to 54 Mbps.

Coded OFDM (COFDM) is a channel coding and modulation scheme which mitigates the adverse effects of fading by using wideband multicarrier modulation combined with time interleaving and a convolutional error correcting code. A guard interval is inserted at the transition between successive symbols to absorb the intersymbol interference created by the time domain spread of the mobile radio channel. The decoding process is performed using differential demodulation in conjunction with a hard decision Viterbi decoder.

The objective of simulating the physical layer of IEEE 802.11a using MATLAB has been successfully achieved in this thesis. The simulation results show a COFDM system capable of indoor environment communications in the presence of known multipath and noise conditions. Further discussions relating to specific test phases are presented below.

Test phase 1 validated a functionally correct model, as there were an absence of errors in the sink message with no channel included. This indicated that at least functionally all system sub-blocks within the transmitter and the receiver were operating correctly according to design, and no obvious design flaws existed due to inaccurate m-file construction. Test phase 2 carried the functional verification one step further by also including complete system model 1 simulations (with AWGN only). This test permitted performance curve comparisons to the work reported by *Louis Thibault and Minh Thien Le, IEEE 1997*. Results of system simulations indicated that system model 1

performance is approximately 0.9dB (at Pb=10^{-3}) and 1.05dB (at Pb=10^{-2}) worse than *Louis Thibault and Minh Thien Le, IEEE 1997* most likely due to hard decision decoding.

Test phase 3 simulation using the channel 2 model (multipath channel only) exclusively demonstrated the effects of multipath on the received signal and the corresponding sink message array error event manifestations. As expected, frequency selective fading occurred as well as partial flat fading. As anticipated, these plots demonstrated constructive and destructive interference due to channel multipath distortions, as evident by the distinguishing peak and valley apparent in the received signal magnitude plots.

Test phase 4 involved comprehensive testing of a complete system simulations using a combined model (AWGN and multipath) to generate corresponding system performance curves. In comparison to test phase 2 (AWGN only), the results shown that extra dB are required to combat the multipath effect. The extra dB needed is between 1.80 to 1.85dB at 10^{-2} probability and between 2.75 to 2.80dB at 10^{-3} probability. The results obtained also show that the COFDM configuration is immune to Doppler shift of 5 to 15 Hz. Since our COFDM configuration uses only 48 tones, it offers good Doppler immunity as the frequency spacing is larger. The above Doppler frequencies used are all less than 1m/s which are good representation of human's walking speed in an indoor environment. Hence we can further deduce that this COFDM configuration is robust enough to withstand the indoor mobility requirements.

The COFDM configuration was further examined with DBPSK modulation. As expected, the results shown that DBPSK required less E_b/N_o then DQPSK. Under the influence of AWGN and multipath, the DBPSK modulations show that the E_b/N_o required for links 1 to 3 simulations are similar.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The IEEE 802.11a WLAN standardization group has selected Orthogonal Frequency Division Multiplexing (OFDM) as the basis for the new 5 GHz standard, targeting a range of data rates from 6 up to 54 Mbps. The objective of this thesis is to develop a simulation of the IEEE 802.11a's physical layer using MATLAB, and to study the effects of multipath on its symbol and bit error performance. The performance curves obtained in this thesis have the potential for high visibility and impact in several operational projects [1]. The increasing prevalence of WLAN, both within the Defense establishment and in the public domain, underscores the need for a simulation of this kind. The results obtained from this thesis can be included into the Radio pipeline of OPNET simulation package. The OPNET version 7 comes with an IEEE 802.11 model, and it can be modified to function as an IEEE 802.11a WLAN. Hence, the performance of this newly proposed IEEE 802.11a WLAN protocol in either an office or submarine environment can be completely analyzed.

A. BACKGROUND

The need for mobile communications and computing, combined with the explosive growth in demand for Internet access, suggest a very promising future for wireless data services. Every day, more and more applications are found that can benefit from wireless networks. The wide range of applications varies from home and small office uses to military uses.

Since ships have limited personnel, it is vital to increase the productivity of every crewmember onboard. Accurate, timely communications between the casualty scene, different stations around the ship, and damage control central have always been of the utmost importance when combating shipboard casualties. Similarly, the efficient dissemination of accurate information is also critical to the success of a submarine's operations. To tap the benefits of wireless networking, the NAVSEA New Attack

Submarine (NSSN) program has identified two key areas, damage control communications and watchstander logs, for productivity improvement by deploying wireless local area networks (WLANs) onboard submarines [1].

B. WIRELESS LOCAL AREA NETWORKS – IEEE 802.11A

Since the beginning of the 1990s, WLANs for the 900 MHz, 2, 4, and 5 GHz industrial, scientific, and medical (ISM) bands have been available based on a range of proprietary products. In June 1997 the IEEE adopted the first standard for WLANs, IEEE Std 802.11-1997. This standard was revised in 1999. The standard specifies both medium access control (MAC) procedures and three different physical layers (PHY). There are two radio-based PHYs using the 2.4GHz band. The third PHY uses infrared light. All PHYs support a data rate of 1 Mbps and optionally 2 Mbps [2].

A second IEEE 802.11a working group was formed to standardize yet another PHY option, which offers higher data rates in the 5.2 GHz band. This development was motivated by the U.S. Federal Communications Commission amendment to part 15 of its rules. The amendment made available 300 MHz of spectrum in the 5.2 GHz band, intended for use by a new category of unlicensed equipment called unlicensed national information infrastructure (UNII) devices.

In July 1998, the IEEE 802.11a standardization group decided to select orthogonal frequency-division multiplexing (OFDM) as the basis for their new 5GHz standard, targeting a range of data rates from 6 up to 54 Mbps. This new standard is the first to use OFDM in packet-based communications; the use of OFDM was previously limited to continuous transmission systems like digital audio broadcasting (DAB) and digital video broadcasting (DVB). Following the IEEE 802.11 decision, in Europe the High-Performance LAN (HIPERLAN) type 2, and in Japan the Multimedia Mobile Access Communication (MMAC) also adopted OFDM for their PHY standards. The three bodies have worked in close cooperation since then to ensure that differences between the various standards are kept to a minimum, thereby enabling the manufacturing of equipment that can be used worldwide [3].

C. OFDM

In indoor radio communication, special propagation problems arise due to the highly reflective, shadowing environment. Radio signals propagate via multiple paths which differ in amplitude, phase and delay time[4]. If the symbol period gets shorter than the root mean square delay spread of the radio channel, significant distortion and intersymbol interference occurs in the receiver signals. Equalization in this case is complicated, complex and expensive.

A totally different way to overcome the problem of multipath fading is the multicarrier approach. The given system bandwidth is divided into an appropriate number of subbands each of which is modulated with a low data rate modulation, corresponding to a long symbol period. OFDM is a special case of multicarrier modulation, where a guard time is inserted between consecutive symbols. This guard interval avoids intersymbol interference and if differential modulation schemes are applied to the subcarriers, no equalization is required at all.

D. RELATED WORK

In [14], *Thibault and Le* had configured a COFDM system based on DQPSK modulation, convolutional code of ½ with constraint length of 7, and the decoding process is performed using differential demodulation in conjunction with a soft decision Viterbi decoder. The BER vs Eb/No curves were simulated in the Additive White Gaussian Noise (AWGN) channel. This research provides a basis for validating the COFDM simulation of this thesis and acts a starting point for further comparative analysis.

In [17], *David V. Roderick* explored the application of COFDM toward a high-data-rate line-of-sight maritime communications modem. The modem model was simulated in MATLAB, and it was used to investigate the feasibility and reliability of digital communications system for ship-to-ship, ship-to-shore, and ship-to-relay type connectivity. This simulation work acted as a starting point for the simulations of this thesis.

E. ORGANIZATION OF THE STUDY

This thesis is organized as follows :

Chapter II provides an overview of the system configuration, and offers the reader a detailed concept description of the coded OFDM model used in this thesis. Chapter III discusses the noise channels. It provides descriptions of the additive white Gaussian noise (AWGN) channel multipath channel. Chapter IV focuses on the MATLAB COFDM System models. It covers the implementations of the transmitter, receiver and noise channels. In Chapter V, the MATLAB programming and development are described. Three different OFDM receiver models are covered. The three models are namely the noise free model, the AWGN model and the combination of AWGN and multipath model. Chapter VI offers the system simulation methodology and the test results. The test plan consists of four different phases aim to verify proper integration of various sub-blocks and validation of system model. Finally, Chapter VII summarizes this thesis research and offers a road map for future researches.

II. MODEL DESCRIPTIONS

A. OVERVIEW OF CONFIGURATION

The basic configuration of a coded OFDM model is shown in Figure 1. The data to be transmitted are first encoded using a convolutional encoding technique, next the data are interleaved via conventional block interleaver. The purpose of the block interleaver and convolutional encoder is to improve the symbol/bit error rate performance. The data to be transmitted are divided among several subcarriers, and the subcarrier signals, which are modulated by the divided data, are transformed into a time domain signal using an inverse fast Fourier transform (IFFT). The IFFT output signal is then formed into an OFDM symbol by extending the IFFT output cyclically. This cyclic extension is often called the guard interval. At the receiver side, the inverse operation is performed to demodulate the received signal. The guard interval is removed from the received signal and the resulting signal is demultiplexed into subcarrier signals by the FFT. The subcarrier signals are detected and the results combined to yield the received data. The detailed descriptions of the above mentioned main components are provided in the following sections.

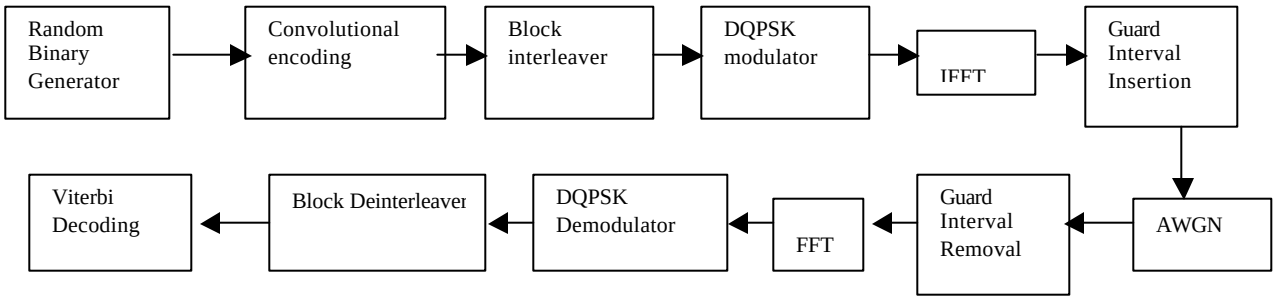


Figure 1. Basic Configuration Of a Coded OFDM Model.

B. CONVOLUTIONAL ENCODING

The Convolutional Encoder receives the messages from the Random Binary Generator and encodes them into codewords. Convolutionally encoding the data is accomplished using a shift register and associated combinatorial logic that performs modulo-two addition. A shift register is merely a chain of flip-flops wherein the output of the nth flip-flop is tied to the input of the (n+1)th flip-flop. Every time the active edge of the clock occurs, the input to the flip-flop is clocked through to the output, and thus the data are shifted over one stage. The combinatorial logic is often in the form of cascaded exclusive-or gates.

The convolutional encoder in this COFDM model is depicted below [5]:

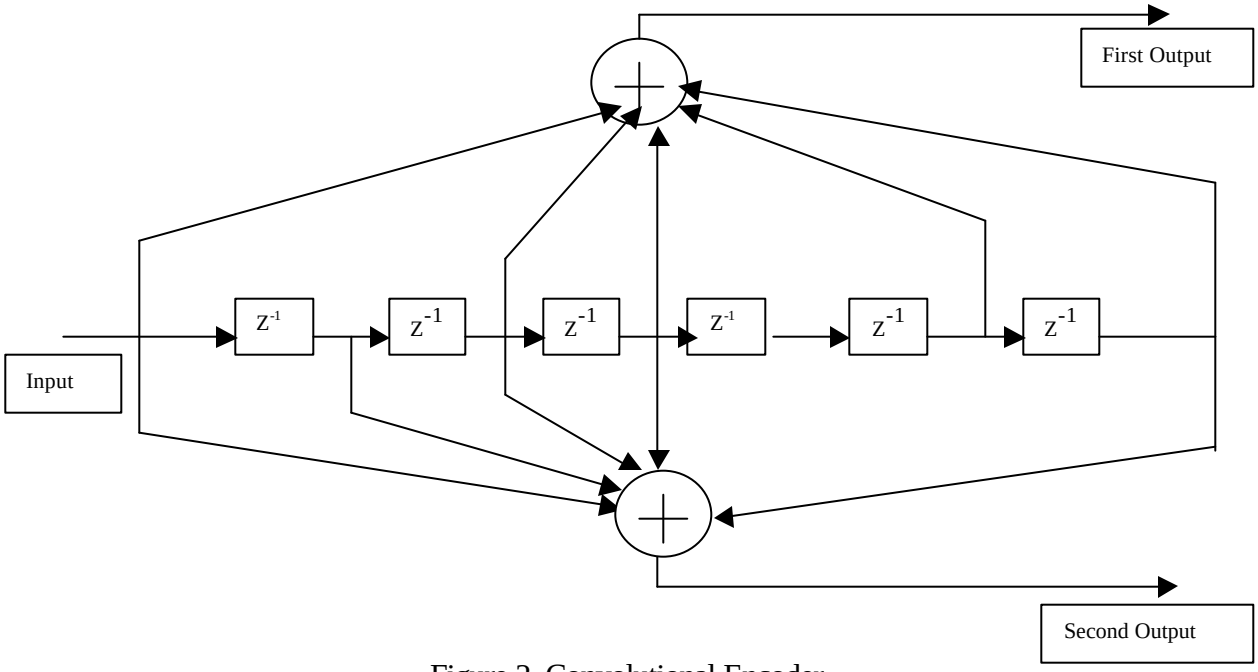


Figure 2. Convolutional Encoder.

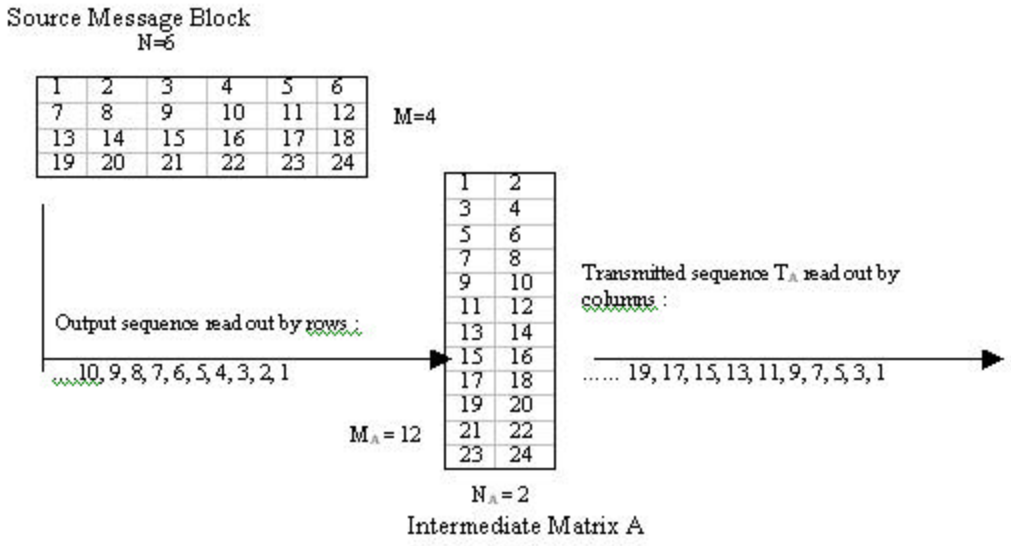
As shown in Figure 2, each summing node represents modulo-two addition. Each box marked Z^{-1} represents a memory register that holds the input values from previous sample times. Since there are six memory registers, the output at a given time depends on seven input values, including the current one. Thus the *constraint length* of the code is 7. Since the code has one input and two outputs, the code rate is ½.

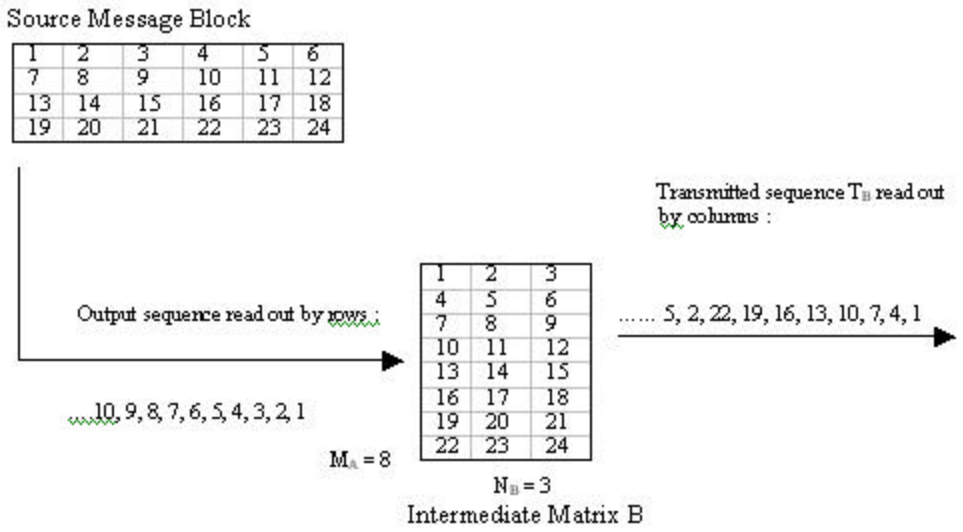
A pair of octal numbers called the *code generator* indicates the connections from the memory registers to the modulo-two summing nodes. The pair [133_8 171_8] (i.e. [1011011_2 1111001_2]) describes the encoder in the figure.

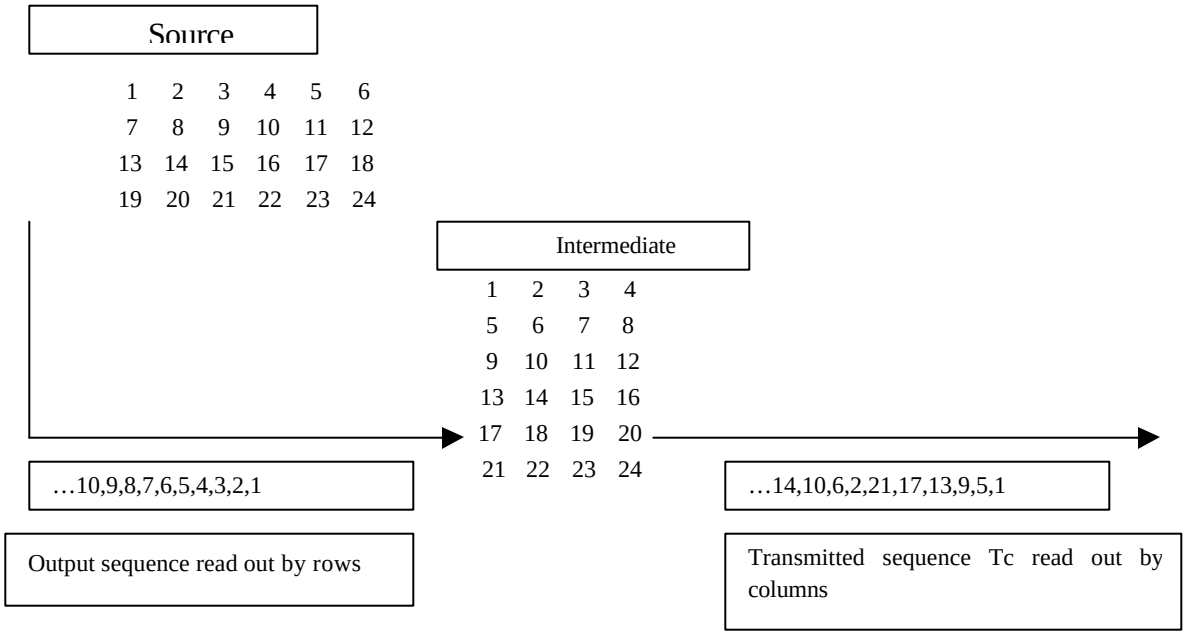
C. BLOCK INTERLEAVER

A block interleaver accepts the coded symbols in blocks from the encoder, permutes the symbols, and then feeds the rearranged symbols to the modulator. The usual permutation of the block is accomplished by filling the columns of an M-row-by N-column (MxN) array with the encoded sequence. After the array is completely filled, the symbols are then fed to the modulator one row at a time and transmitted over the channel. At the receiver, the deinterleaver performs the inverse operation; it accepts the symbols from the demodulator, deinterleaves them, and feeds them to the decoder. Symbols are entered into the deinterleaver array by rows, and removed by column. Figure 3 illustrates an example demonstrating the effects of interleaving a message prone to burst errors prior to transmission through the channel. In this example, the symbol coded source message block is structured as a M by N matrix, S, with M = 4 rows and N=6 columns and the dimension product of S equal to MxN =24. As part of the interleaving algorithm an intermediate matrix must be temporarily constructed using the symbols taken from S. Therefore, the dimension product of the intermediate matrix, L, (# of column times # of rows) also equal to M x N. Given the value of S for this example, all possible row and column intermediate matrix dimension pairs are : (1, 24), (2, 12), (3, 8), (4, 6), (6, 4), (8, 3), (12, 2), and (24, 1). During the formation and subsequent filling of the intermediate arrays having each of these dimensions, the symbols provided by matrix S are read out row-by-row and into L row-by-row until S is empty. After matrix L becomes full, the individual symbols within are read out column-by-column, representing the transmission sequence. It is evident that effective decorrelation of adjacent errored symbols within the transmitted message sequence depends on selective formation of intermediate matrices using appropriate array dimensions. Varied matrix dimensions tend to space the errors differently throughout the message block after deinterleaving is performed.

Figure 3 supports this example pictorially. It is instructive to note that formation of intermediate arrays with dimension (1, 24) (row vector) and (24, 1) (column vector) are not generally implemented since no effective interleaving occurs. For instructional purposes, this example uses intermediate matrix interleaver dimension pairs : (2, 12), (3, 8) and (4, 6) only. From Figure 3, the dimensions of intermediate matrix A are (12, 2), having 12 rows and 2 column. After being filled completely with the symbol taken from the source message block read in row by row, the transmitted sequence, T_A, is read out of matrix A column by column. During transmission through the channel, hypothetical burst noise occurs corrupting a group of three adjacent symbols in the sequence. Upon reception, the receiver deinterleaves the sequence to reconstruct the original source message. It is apparent from the figure that the burst errors become decorrelated from the group after deinterleaving, becoming isolated non-adjacent symbol errors spaced every other symbol apart.




Source Message Block
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17 18
19 20 21 22 23 24
Output sequence read out by rows :
…10, 9, 8, 7, 6, 5, 4, 3, 2, 1
$M_A = 8$
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
16 17 18
19 20 21
22 23 24
$N_B = 3$
Intermediate Matrix B
Transmitted sequence T_B read out by columns :
…… 5, 2, 22, 19, 16, 13, 10, 7, 4, 1


Source
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17 18
19 20 21 22 23 24
Intermediate
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
21 22 23 24
…10,9,8,7,6,5,4,3,2,1
Output sequence read out by rows
…14,10,6,2,21,17,13,9,5,1
Transmitted sequence Tc read out by columns

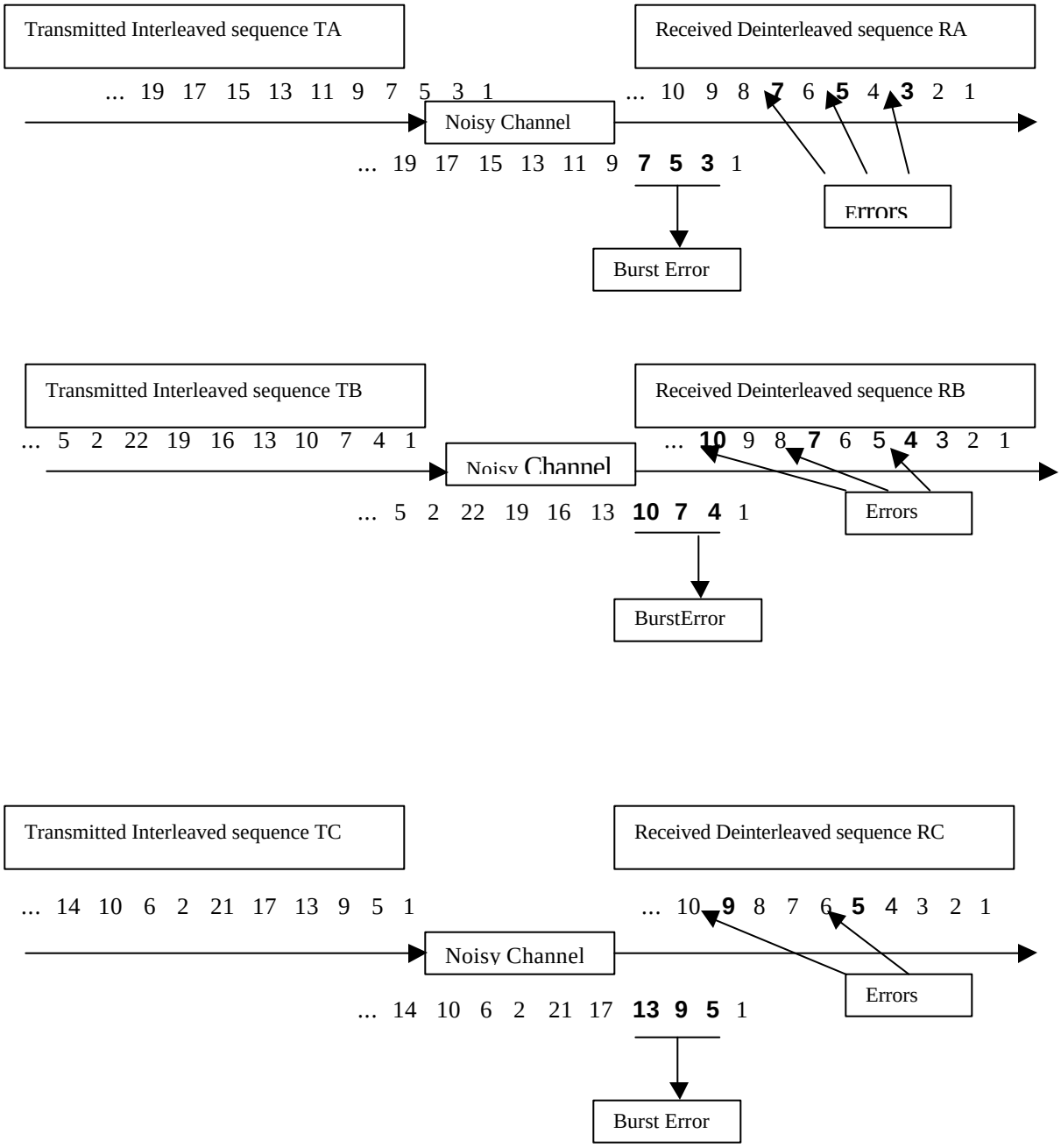


Figure 3. Demonstrating The Effects of Interleaving a Message Prone To Burst Errors.

In a similar example, using intermediate matrix B with 8 rows and 3 columns, the identical channel burst error event once again affects a group of three symbols in the transmitted sequence, T_B. Following deinterleaving in the receiver, the group of contiguous errors become decorrelated forming isolated symbol errors in the received

sequence, R_B, spaced every two symbols apart. Similarly, for the last intermediate matrix C example, following transmission of the interleaved sequence, T_C, through the channel and deinterleaving in the receiver , the group of errored symbols afflicted by burst noise in the channel become singly occurring error events spread out in the received message sequence, R_C, and are spaced every third symbol apart. If this example continued for every possible interleaver intermediate matrix dimension, it becomes apparent that the spacing of isolated errors appearing in deinterleaved message sequences are directly related to the intermediate matrix dimensions [5].

D. DQPSK MODULATION

The term differential PSK (DPSK) refers to the procedure of encoding the data differentially; that is, the presence of a binary one or zero is manifested by the symbol's similarity or difference when compared to the preceding symbol. The term DPSK is often classified as noncoherent because it does not require a reference in phase with the received carrier [5].

With noncoherent systems, no attempt is made to determine the actual value of the phase of the incoming signal. Therefore, if the transmitted waveform is

$$Si = \sqrt{\frac{2E}{T}} \cos\,[\omega_o t + q_i(t)] \qquad 0 \le t \le T \qquad i = 1,\ldots\ldots M \tag{2-1}$$

the received signal can be characterized by :

$$r(t) = \sqrt{\frac{2E}{T}} \cos\,[\omega_o t + q_i(t) + \alpha] + n(t) \qquad 0 \le t \le T \qquad i = 1,\ldots\ldots M \tag{2-2}$$

Where α is an arbitrary constant and is typically assumed to be a random variable uniformly distributed between zero and $2p$, and *n(t)* is an AWGN process.

For coherent detection, matched filters are used; for noncoherent detection, this is not possible because the matched filter output is a function of the unknown angle.

However, if we assume that α varies slowly relative to two period times (2T), the phase difference between two successive waveforms $\theta_j(T1)$ and $\theta_k(T2)$ is independent of α, that is,

$$[q_k(T_2) + a] - [q_j(T_1) + a] = q_k(T_2) - q_j(T_1) = f_i(T_2) \qquad (2\text{-}3)$$

The basis for differentially coherent detection of differentially encoded PSK (DPSK) is as follows. The carrier phase of the previous signaling interval can be used as a phase reference for demodulation. Its use requires differential encoding of the message sequence at the transmitter since the information is carried by the difference in phase between two successive waveforms. To send the ith message (i= 1,2,….M), the present signal waveform must have its phase advanced by $f_i = 2p$ i/M radians over the previous waveform. The detector, in general, calculates the coordinates of the incoming signal by correlating it with locally generated waveforms such as $\sqrt{\frac{2}{T}}\cos w_o t$ and $\sqrt{\frac{2}{T}}\sin w_o t$. The detector then measures the angle between the currently received signal vector and the previously received signal vector.

In general, DPSK signaling performs less efficiently than PSK, because the errors in DPSK tend to propagate (to adjacent symbol times) due to the correlation between signaling waveforms. One way of viewing the difference between PSK and DPSK is that the former compares the received signal with a clean reference; in the latter, however, two noisy signals are compared with each other. One might say that there is twice as much noise associated with DPSK signaling compared to PSK signaling. It is estimated that DPSK manifests a degradation of approximately 3 dB when compared with PSK; this degradation decreases rapidly with increasing signal-to-noise ratio. The trade-off for this performance loss is reduced system complexity [5].

The use of coherent demodulation (PSK) will give better performance if the channel is not varying too much. For mobile reception, the channel response may vary rapidly in phase, and so the potential benefits of coherent demodulation (PSK) are lost in

the implementation [6]. It is easier to implement a DPSK system than a PSK system, since the DPSK receiver does not need phase synchronization. For this reason, DPSK, although less efficient than PSK, is the preferred choice.

E. OFDM

In a traditional serial data digital communication system, data is sent as a serial pulse train of information symbols. During the sequence transmission of each symbol through the channel, the symbol frequency spectrum is allowed to occupy the entire available bandwidth. However, in a multipath environment (i.e. highly reflective and shadowing indoor communication or a maritime environment with scattered reflections from the ocean surface) the signal envelope fluctuates. The time dispersion nature of the multipath channel also causes adjacent symbols of the serial stream to interfere when the symbols are short compared to the time spread [7].

A parallel communication system differs from the serial counterpart by allowing the simultaneous transmission of several sequential data streams using much longer symbols. At any instance in time, there are many data elements (symbol) being transmitted through the channel. With this type of system, the individual spectrums of each data symbol occupy only a small portion of the overall available bandwidth. This approach is advantageous in spreading out the frequency-selective fade over many different symbols. Thus, instead of there being a high concentration of errors with several adjacent symbols being completely destroyed by the fade, the errors are spread out over many symbols and appear less bursty. In this situation, precise reconstruction of a majority of the symbols is possible even without the addition of error correcting codes. Additionally, in a parallel system, by partitioning the entire bandwidth into multiple non-overlapping frequency sub-bands (sub-channels), equalization of each sub-channel is much easier than the serial system because the symbols are now much longer than the time dispersion of the channel, which greatly reduces the effects of ISI.

The approach to implementing a parallel communications system is done in different ways. In a classical parallel data system using conventional FDM technology

(Figure 4), the total signal frequency bandwidth is partitioned into N non-overlapping sub-channels and are frequency-division multiplexed for transmission.

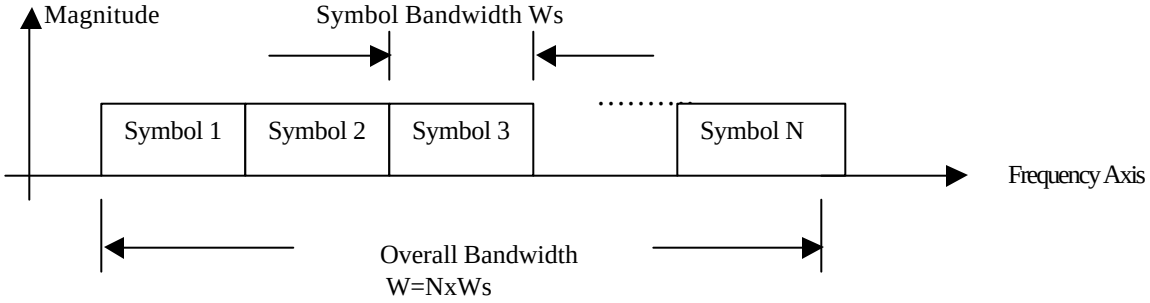


Figure 4. Ideal Frequency-Division Multiplexing Spectrum.

At the receiving end, separation of the sub-band traditionally is accomplished by a bank of bandpass filters. However, due to the roll-off effect of physically filters, the actual bandwidth of each sub-channel must be further widened. Sufficient guard bands must be inserted in the frequency spectrum between adjacent sub-channels to permit effective filtering without in-band signal attenuation and adjacent band signal interference. This method, with the addition of guard bands, does not offer the best possible spectrum efficiency (Figure 5) since now the overall bandwidth is lengthened by multiple guard bands that do not carry any useful information. [8].

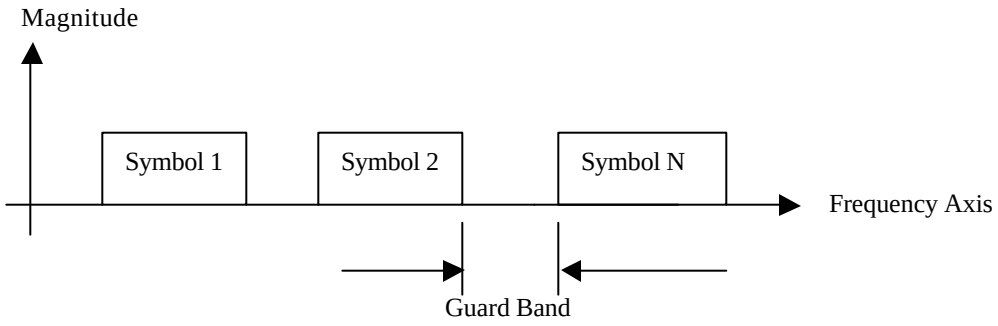


Figure 5. Additional Guard Band In Frequency Spectrum.

An alternative to traditional FDM is a system that uses the discrete Fourier transform (DFT) to modulate and demodulate parallel data. Using the DFT in the transmitter, the individual sub-channel spectra can be represented with sine functions

which are not band-limited. Multiplexing of the sub-channels is accomplished by base-band processing instead of bandpass filtering.

One such technique which uses the DFT for implementation is Orthogonal Frequency Division Multiplexing (OFDM), which is defined as a form of multi-carrier modulation where the carrier spacing is carefully selected so that each sub-carrier (tone) is orthogonal to the other sub-carriers. In order for a signal set to be orthogonal, any pair of sub-carriers must have a frequency separation of a multiple of 1/Ts [8]. OFDM differs from traditional FDM by allowing the OFDM spectrum of individual orthogonal subcarriers to mutually overlap; thus, a more optimum spectrum efficiency is gained over FDM. With the inclusion of coherent detection at the receiver and the use of orthogonal subcarrier tones separated by the reciprocal of the signaling element duration, independent separation of the multiplexed tones is possible, specifically by using the DFT.

Consider a data sequence (D_0, D_1, D_2,….D_{N-1}), where each D_n is a complex number of the form $D_n=A_n +jB_n$. If a DFT is performed on the sequence, the result is a vector $\underline{d} = (d_0, d_1, d_2, \ldots d_{N-1})$ of N complex numbers with :

$$d_m = \sum_{n=0}^{N-1} D_n \exp(-j(\frac{2pnm}{N}) = \sum_{n=0}^{N-1} D_n \exp(-j(2pf_n t_m)), \quad \text{m=0, 1, 2, ….N-1,} \quad (2\text{-}4)$$

Where $f_n \approx \frac{n}{N\Delta t}$,
$t_m \approx m\Delta t$,
$\Delta t \approx \frac{T_s}{N}$,

and T_s is an arbitrary chosen duration of the serial data sequence D_n [7]. Taking the real part only of the $\underline{d}$ vector, we get the following components :

$$y_m = \sum_{n=0}^{N-1} A_n \cos(2pnf_n t_m) + \sum_{n=0}^{N-1} B_n \sin(2pnf_n t_m) \qquad \text{m=0, 1, 2, …N-1} \qquad (2\text{-}5)$$

Applying these components to an ideal low-pass filter with cutoff frequency $\frac{f_s}{2} = \frac{1}{2\Delta t}$, we now obtain the frequency division multiplexed signal :

$$y(t) = \sum_{n=0}^{N-1} A_n \cos(2pnf_n t) + \sum_{n=0}^{N-1} B_n \sin(2pnf_n t) \qquad 0 \le t \le T_s \qquad (2\text{-}6)$$

As an illustration of a general OFDM based communication system using the orthogonality principle, Figure 6 represents a block diagram of major system components with substitutions of more efficient fast Fourier transform (FFT) and inverse fast Fourier transform (IFFT) algorithms to reduce the number of operations from N^2 in the DFT down to approximately $\frac{N}{2}\log_2 N$ for the radix two FFT [9].

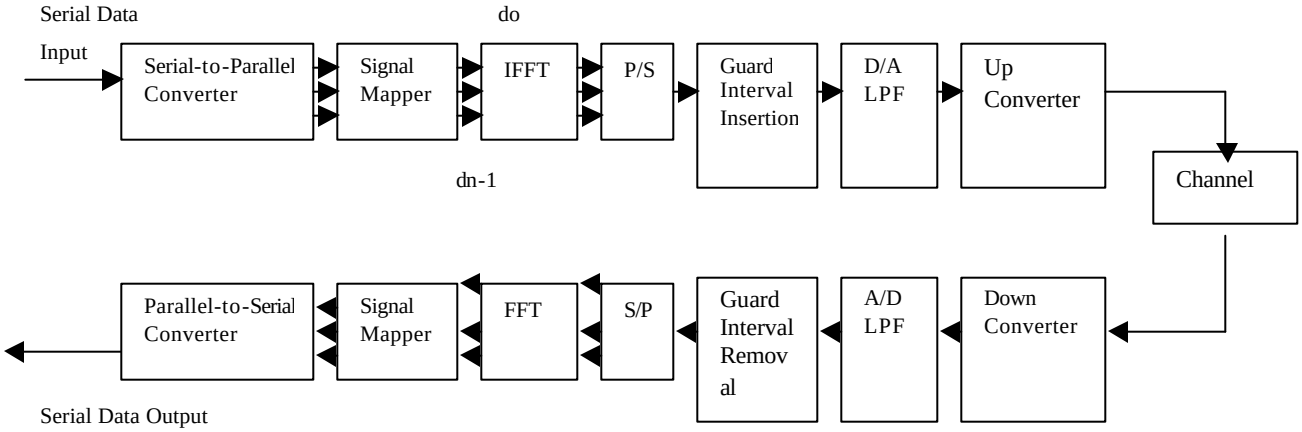


Figure 6. FFT-based OFDM System.

Initially, the incoming serial data bit stream is grouped to form symbols, q bits long, in preparation for a M-ary digital modulation scheme, where $M=2^q$. Each symbol passes through a signal constellation mapper, such as 4-phase shift keyed (4-PSK) for example (for this case, $q=\log_2 M=\log_2 4=2$), to generate a complex modulation value,{DN},corresponding to a particular 2-bit symbol. The sequence of complex modulation values are converted from serial to parallel format by a multiplexer to form a block size of N symbols, where each member of N corresponds to a baseband fashion by

the IFFT performing the mapping into the time domain. Finally a multiplexer converts from parallel format to a serial data stream suitable for up conversion and RF transmission. Before the up conversion process can be accomplished, an analog-to-digital (A/D) converter is used to convert the discrete values to the analog equivalent and perform low-pass filtering. After transmission through the channel, the OFDM receiver portion of the system performs the inverse process of the transmitter. Specifically, down conversion and low-pass filtering is initially performed to recreate the baseband transmitted signal. The baseband serial data stream is converted to parallel forming N paths, which are fed to an FFT block. The N-point FFT operation recovers the complex modulation values, allowing the inverse signal mapper to generate the corresponding symbol bit pattern. The q-bit length symbols are multiplexed into a serial data stream to complete the process and recover the original information.

During the signal constellation mapping stage, each data symbol is encoded as a truncated sinusoid within the interval $(0,T_s)$. Signal truncation causes the frequency response of y(t) to be a sinc function. As seen in Figure 7, the spectral shape of an OFDM subchannel contains zero crossings at multiples of $1/T_s$. The other sub-carriers are generated by the IDFT in such a way that their spacing generates a nearly flat overall spectrum with no interference among individual spectra. For example, an OFDM spectrum would be similar to the one depicted in Figure 8. In this figure the orthogonality of the subcarriers is demonstrated by the overlapping of individual subcarrier spectra at their respective zero crossings, thus, the spectra of the individual subchannels are zero at the other subcarrier frequencies.

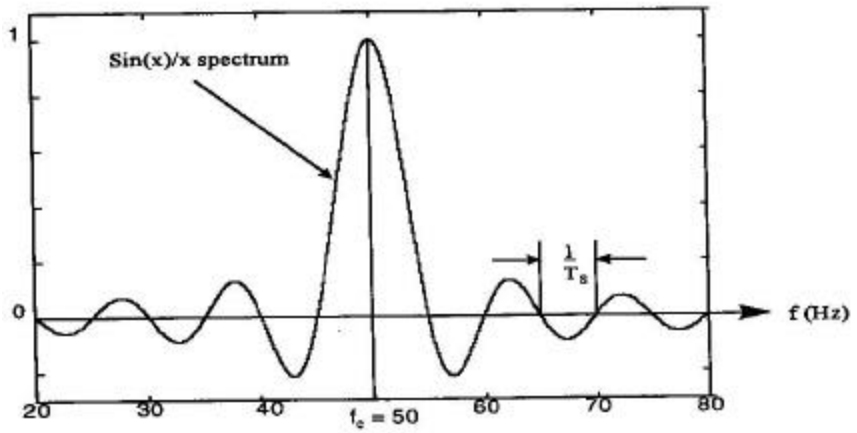


Figure 7. Spectrum For Single Symbol With Length Ts.

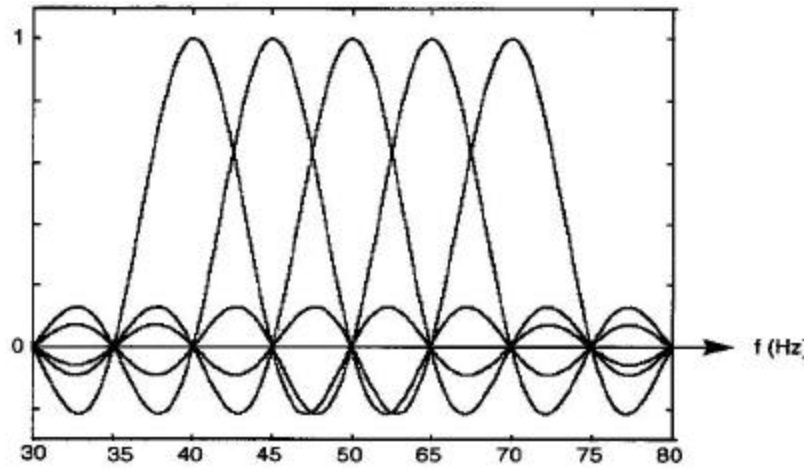


Figure 8. OFDM Spectrum.

As previously mentioned, generation of this orthogonal structure is accomplished by using the IFFT, and assuming a distortionless channel, orthogonality is maintained after transmission with each individual subchannel completely separable by the FFT process in the receiver. Unfortunately, in practice, ideal distortionless channel conditions cannot be guaranteed and are typically nonexistent in actual RF transmission environments. Also, since each OFDM symbol spectrum is not band limited, channel distortions such as multipath cause each subchannel to spread energy into the adjacent subchannels causing intercarrier interference (ICI).

Orthogonal Frequency Division Multiplexing (OFDM) is a modulation method that, like all wireless transmission schemes, encodes data onto a radio frequency signal. Conventional

single carrier transmission schemes like AM/FM (amplitude or frequency modulation) send only one signal at a time using one radio frequency. OFDM sends multiple high-speed signals concurrently on different frequencies. This results in very efficient use of bandwidth, and provides robust communications in the presence of noise, intentional or unintentional interference, and reflected signals that degrade radio communications.

F. GUARD INTERVAL

As mentioned in the above section, the basic principle of OFDM is to split a high-rate data stream into a number of lower-rate streams which are transmitted simultaneously over a number of subcarriers. Since the symbol duration increases for lower-rate parallel subcarriers, the relative amount of time dispersion caused by multipath delay spread is decreased. ISI is eliminated almost completely by introducing a guard time in every OFDM symbol. In the guard time, the OFDM symbol is cyclically extended to avoid intercarrier interference. Figure 9 shows an example of four subcarriers from one OFDM symbol. It can be seen in Figure 9 that all subcarriers differ by an integer number of cycles within the FFT integration time, which ensures orthogonality between the different subcarriers. This orthogonality is maintained in the presence of multipath delay spread, as shown in Figure 9. Because of multipath, the receiver sees a summation of time-shifted replicas of each OFDM symbol. As long as the delay spread is smaller than the guard time, there is no ISI or intercarrier interference within the FFT interval of an OFDM symbol [10].

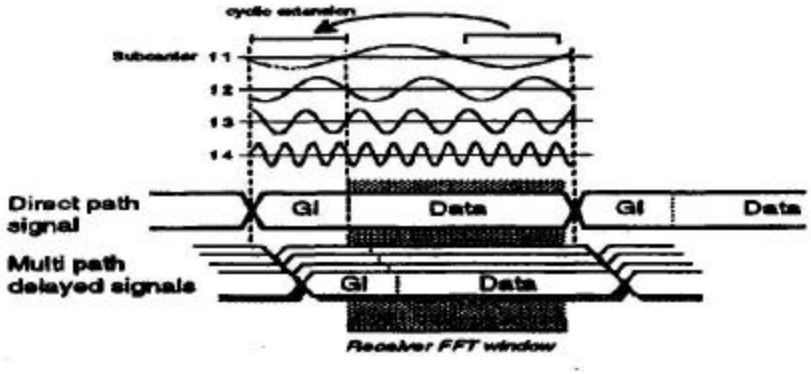


Figure 9. An Effect Of Guard Interval.

Table 1 below lists the main parameters of the IEEE 802.11a OFDM standard. A key parameter which largely determined the choice of the other parameters is the guard interval of 800ns (0.8μs). This guard interval provides robustness to RMS delay spreads up to several hundreds of nanosecond, depending on the coding rate and modulation used. In practice, this means that the modulation is robust enough to be used in any indoor environment, including large factory buildings. It can also be used in outdoor environments, although directional antennas may be needed in this case to reduce the delay spread to an acceptable amount and increase the range.

Data Rate	6, 9, 12, 18, 24, 36, 48, 54 Mbps
Modulation	BPSK, QPSK, 16-QAM, 64-QAM
Coding Rate	½, 2/3, 3/4
Number of subcarriers	48 (without pilot tones)
Number of FFT points	64
OFDM symbol period	4 μs
Guard Interval	800ns
Subcarrier spacing	312.5KHz
-3dB bandwidth	16.6MHz
Channel spacing	20MHz

Table 1. Main Parameters Of The OFDM Standard.

G. VITERBI DECODER

The Viterbi algorithm is a method commonly used for decoding bit streams encoded by convolutional encoders. This algorithm is a maximum-likelihood decoding algorithm, which upon receiving the channel output, searches through the trellis to find the path that is most likely to have generated the received sequence. If hard-decision decoding is used, this algorithm finds the path that is at the minimum Hamming distance from the received sequence, and if soft-decision decoding is employed, the Viterbi algorithm finds the path that is at the minimum Euclidean distance from the received sequence.

1. Code Trellis

For ease of explanation, a simple convolutional encoder with constraint length of 3, and rate ½ is used (Figure 10).

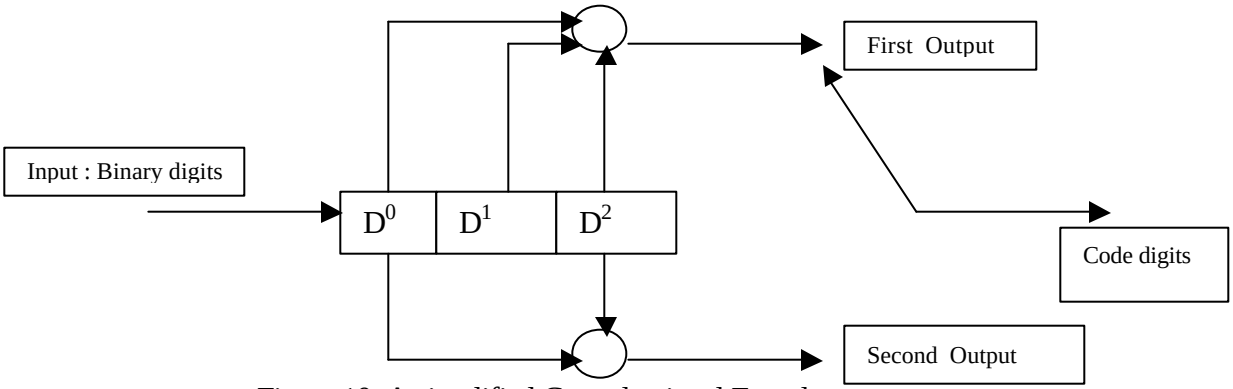


Figure 10. A simplified Convoluational Encoder.

For the rate 1/2 convolutional code presented in Figure 10, the Code Trellis is drawn as shown in Figure 11. Notice that it is simply another way of drawing the state diagram, which is presented on the right hand side.

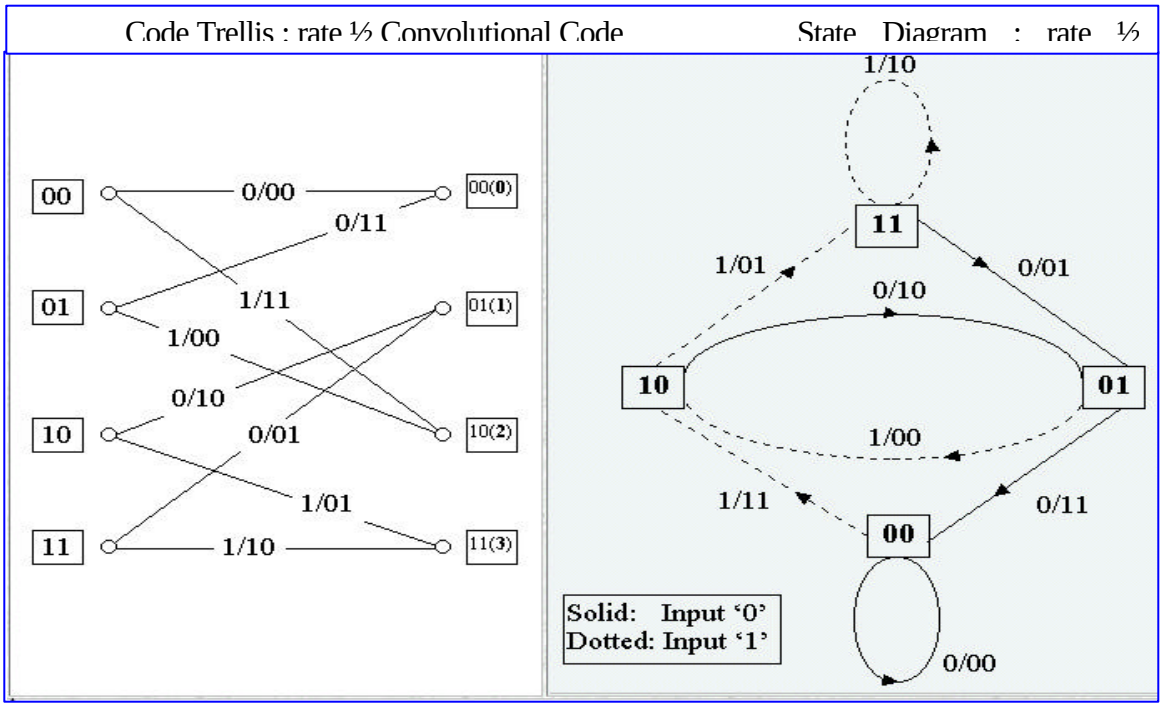


Figure 11. Code Trellis And State Diagram.

The four possible states (00, 01, 10, 11) are labeled 0, 1, 2, 3 (shown in brackets in the code trellis diagram). Notice that there are two branches entering each state, which will be referred to as the *upper* and *lower* branches respectively. For example, the state 01 has an upper branch which comes from the state 10, and a lower branch which comes from state 11. The branch codeword is the codeword associated with a branch. For example, the upper branch entering state 01 has the branch codeword 10. It's labeled 0/10 in the diagram which means that a binary digit 0 input to the encoder in state 10, will output the codeword 10 and move to the state 01.

Using the code trellis, the Viterbi Trellis is drawn as shown in Figure 12. Notice that it is simply a serial concatenation of many code trellis diagrams (ignore the "X", and the highlighted text (yellow) for now). The only important feature at this stage is that the Viterbi trellis consists of many code trellis diagrams. The trellis depth of a Viterbi trellis is the number of code trellis replications used. For example, the trellis depth is 7 in the example below. The diagram below shows the internal operation of the Viterbi decoder using a specific example in which the code sequence 11101111010111 is received from without error.

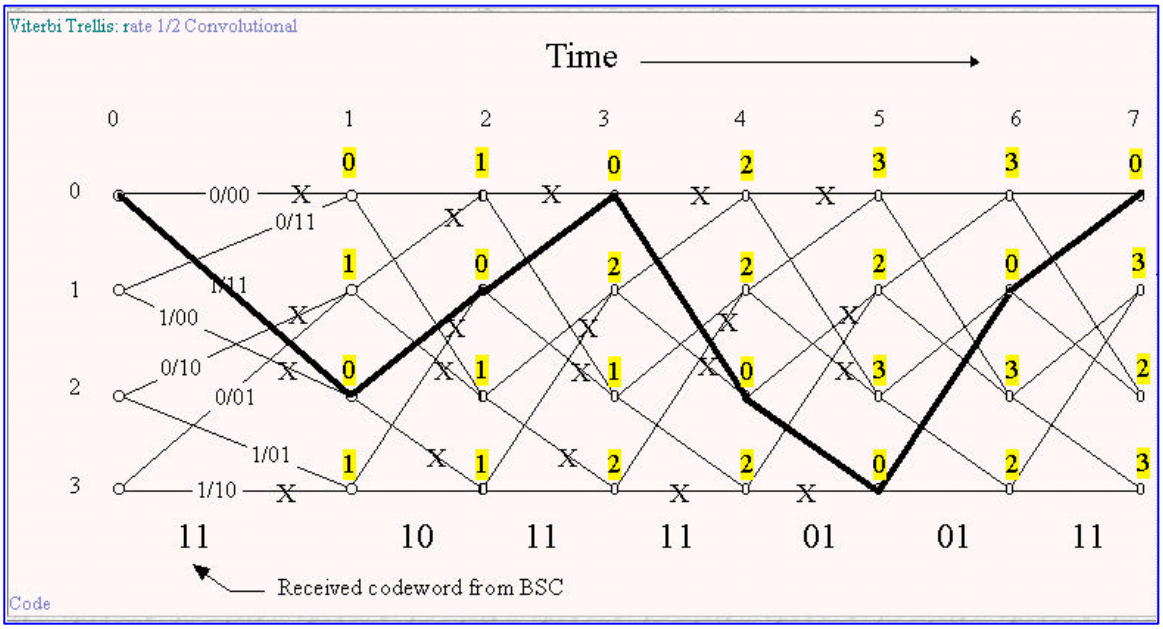


Figure 12. Viterbi Trellis Diagram.

2. Viterbi Algorithm

Any given state in the Viterbi trellis may be identified by the state s and time t. For example the (0, 1) in Figure 13 below represents the state $s = 0$ at time $t = 1$, and (3, 5) represents the state $s = 3$ at time $t = 5$. These states are shown in Figure 13 below so that we can relate them to the main Viterbi trellis diagram in Figure 12.

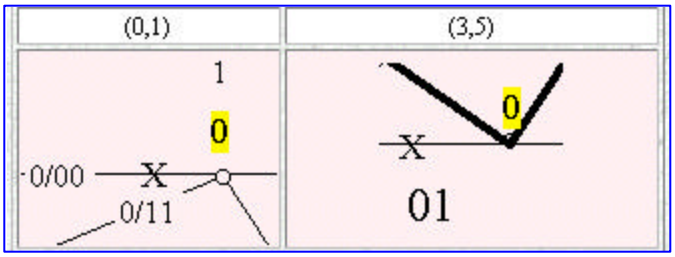


Figure 13. Extract Of Viterbi Trellis Diagram In Figure 12.

Let the metric for a state s at time t be represented by m(s, t). A metric is just a number. This will become clear very shortly. For example, for the two states shown above, the metrics are shown in highlighted, yellow text. Thus $m(0, 1) = 0$ and $m(3, 5) = 0$. At time $t = 0$, we initialize all state metrics to zero (i.e. $m(0,0) = m(1,0) = m(2,0) = m(3,0) = 0$). By setting each state metric to zero, we are taking into account that the encoder may have started in any of the possible states. This is typically the case because even though the encoder does in fact start in the all-zero state, the transmitted codeword sequence may have been segmented and sent as a series of packets. In this case, the starting state of any given segment cannot be assumed to be the all-zero state. If however, we know that the encoder started in the all-zero state for the codeword sequence we are decoding, then for the first code trellis, we need only calculate the metrics which geminate from the state $s = 0$ at time $t = 0$. For example, for the above convolutional code, you need only calculate the metrics $m(0,1)$ and $m(2,1)$ within the first code trellis.

Let the hamming distance for the upper branch entering a state s at time t be HD_upper (s, t), and the hamming distance for the lower branch be HD_lower (s, t). The

Hamming distance is the number of differences between the received codeword and the branch codeword.

The Viterbi Trellis shown in Figure 12 is analyzed in the following steps :

Step 1. At time *t*, for a given state *s*, compare the received binary codeword with each branch codeword entering this state to calculate HD_upper (*s*, *t*) and HD_lower (*s*, *t*). For example, HD_upper (0, 1) = 2 and HD_lower(0,1) = 0.

Step 2. Calculate y_up = HD_upper (*s*, *t*) + m(*s**, *t-1*), where *s* is the state at time *t*, and *s** is the pervious state at time (*t*-1) for a given branch. For example, for the first state *s* = 0 at *t* = 1, y_up = HD_upper (0, 1) + m(0,0) = 2 + 0 = 2.

Step 3. Calculate y_low = HD_lower (*s*, *t*) + m(*s**, *t-1*) For example, for the first state *s* = 0 at *t* = 1, y_low = HD_lower (0, 1) + m(1,0) = 0 + 0 = 0.

Step 4. Identify the surviving branch entering the state at time *t* as follows: Choose upper branch as the survivor if y_up < y_low, and let y_final = y_up. Otherwise choose the lower branch, and let y_final = y_low. If y_up = y_low, then randomly select any branch as the survivor. For example, for the first state *s* = 0 at *t* = 1, y_final = y_low = 0.

Step 5. The branch which does NOT survive is marked with an "X". Only one branch survives per state (or node on the trellis). These X's are only shown in the diagram above up to time *t* = 2. For example, for the first state *s* = 0 at *t* = 1, the upper branch is marked with an "X". This means that this branch does not survive. Only the lower branch entering the state 00 survives.

Step 6. Set the state metric m(*s*,*t*) = y_final. The final metric for each state is shown in Yellow text in the above diagram. For example, for the first state *s* = 0 at *t* = 1, m(0,1) = y_final = 0.

Step 7. Repeat steps 1 to 6 until we reach the end of the Viterbi trellis at time 7. Of course we must determine the metrics m(s,1) first before we can calculate m(s, 2).

Step 8. From all final state metrics [$m(0,7)$ $m(1,7)$ $m(2,7)$ $m(3, 7)$], choose the minimum metric, and trace back the path from this state. In the above example this trace back path is shown as a solid black line, which starts from state $s = 0$ at time $t = 7$, and ends at state $s = 0$ at time $t = 0$.

Step 9. Output the information binary digits which correspond to branches on this trace back path.

3. Metrics

Referring back to the Viterbi Trellis diagram in Figure 12, notice that if we trace back the path which starts at $s = 2$, $t = 5$, the codewords on that trace-back path are as shown in Table 2 below in the first row. Note that at this state, the metric $m(2, 5) = 3$.

Codeword sequence on trace back path from $s = 2$, $t = 5$	11 11 10 11 11
Codeword sequence received from channel	11 10 11 11 01
Hamming distance between these two sequences	3

Table 2. Codewords On The Trace-back Path.

A total cumulative metric $m(2, 5) = 3$ means that the codeword sequence on a path traced back from this state differs with the received codeword sequence in 3 positions.

Hence we select the trace-back path from time $t = 7$ based on which state has the minimum metric. This is because we want to select a codeword sequence within the trellis, which is as close as possible to the received codeword sequence from the channel. i.e. Maximum likelihood decoding [5].

In this chapter we discussed the transmission and reception components of COFDM signals. The next chapter will discuss the communications channel and aspects important to analysis of COFDM.

THIS PAGE INTENTIONALLY LEFT BLANK

III. WIRELESS CHANNEL CHARACTERISTICS

A. NOISE IN COMMUNICATION SYSTEMS

The term noise refers to unwanted electrical signals that are always present in electrical systems. The presence of noise superimposed on a signal tends to obscure or mask the signal. It limits the receiver's ability to make correct symbol decisions, and thereby limits the rate of information transmission. Good engineering design can eliminate much of the noise or its undesirable effect through filtering, shielding, the choice of modulation, and the selection of an optimum receiver site. However, there is one natural source of noise, called *thermal or Johnson noise*, that cannot be eliminated. Thermal noise is caused by the thermal motion of electrons in all dissipative components – resistors, wires, and so on. The same electrons that are responsible for electrical conduction are also responsible for thermal noise [5].

We can describe thermal noise as a zero-mean Gaussian random process. A Gaussian process, $n(t)$, is a random function whose value, n, at any arbitrary time, t, is statistically characterized by the Gaussian probability density function, $p(n)$:

$$p(n) = \frac{1}{s\sqrt{2p}} \exp[-\frac{1}{2}\left(\frac{n}{s}\right)^2] \qquad (3\text{-}1)$$

where s^2 is the variance of n. The normalized or standardized Gaussian density function of a zero-mean process is obtained by assuming that s =1. This normalized pdf is shown sketched in Figure 14.

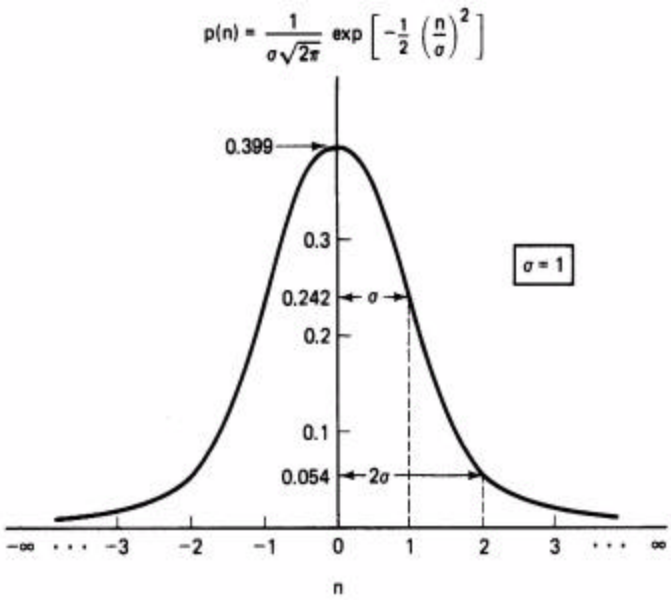


Figure 14. Normalized (s =1) Gaussian Probability Density Function.

A random signal is often represented as the sum of a Gaussian noise random variable and a dc signal :

$$z = a + n \tag{3-2}$$

Where z is the random signal, a the dc component, and n the Gaussian noise random variable. The pdf $p(z)$ is then expressed as

$$p(z) = \frac{1}{s\sqrt{2p}} \exp[-\frac{1}{2}\left(\frac{z-a}{s}\right)^2] \tag{3-3}$$

where, as before, s^2 is the variance of n. The Gaussian distribution is often used as the system noise model because of a theorem, called the central limit theorem, which states that under very general conditions the probability distribution of the sum of j statistically independent random variables approaches the Gaussian distribution as j $\rightarrow \infty$, no matter what the individual distribution functions may be. Therefore, even though individual noise mechanisms might have other than Gaussian distributions, the aggregate of many such mechanisms will tend toward the Gaussian distribution.

B. WHITE NOISE

The primary spectral characteristic of thermal noise is that its power spectral density is the same for all frequencies of interest in most communication systems. In other words, a thermal noise source emanates an equal amount of noise power per unit

bandwidth at all frequencies – from dc to about 10^{12} Hz. Therefore, a simple model for thermal noise assumes that its power spectral density $G_n(f)$ is flat for all frequencies, as shown in Figure 15, and is denoted as follows :

$$G_n(f) = \frac{No}{2} \quad \text{watts/hertz} \tag{3-4}$$

Where the factor of 2 is included to indicate that $G_n(f)$ is a two-sided power spectral density. When the noise power has such a uniform spectral density, we refer to it as white noise. The adjective "white" is used in the sense that white light contains equal amounts of all frequencies within the visible band of electromagnetic radiation.

The autocorrelation function of white noise is given by the inverse Fourier transform of the noise power spectral density denoted as follows :

$$R_n(t) = \Im^{-1}\{G_n(f)\} = \frac{No}{2}d(t) \tag{3-5}$$

Thus the autocorrelation of white noise is a delta function weighted by the factor $N_o/2$ and occurring at $\tau = 0$, as seen in Figure 15(b). Note that $R_n(t)$ is zero for $\tau \neq 0$; that is, any two different sample of white noise, no matter how close together in time they are taken, are uncorrelated.

The delta function in equation 1 means that the noise signal, $n(t)$, is totally decorrelated from its time-shifted version, for any $\tau > 0$. Equation 1 indicates that any two different samples of a white noise process are uncorrelated. Since thermal noise is a Gaussian process and the samples are uncorrelated, the noise samples are also independent. Therefore, the effect on the detection process of a channel with *additive white Gaussian noise* (AWGN) is that the noise affects each transmitted symbol independently. Such a channel is called a *memoryless channel*. The term "additive" means that the noise is simply superimposed or added to the signal – that there are no multiplicative mechanisms at work [5].

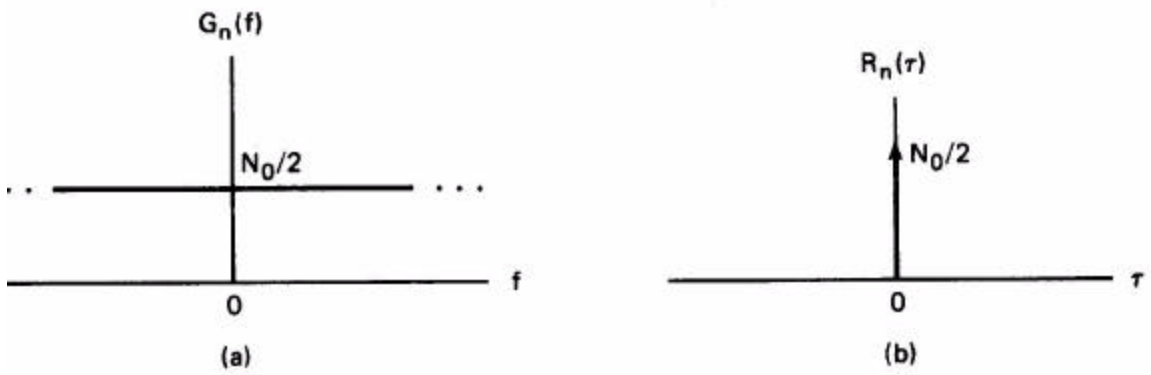


Figure 15. (a) Power Spectral Density Of White Noise. (b) Autocorrelation Function Of White Noise.

C. MULTIPATH

Multipath is one of the performance concerns for indoor IEEE 802.11 WLAN systems. Multipath occurs when the direct path of the transmitted signal is combined with paths of the reflected signal paths, resulting in a corrupted signal at the receiver, as show in Figure 16. The delay of the reflected signals (measured in microsecond (μsec) in this thesis) is commonly known as delay spread. Delay spread is the parameter used to characterize multipath.

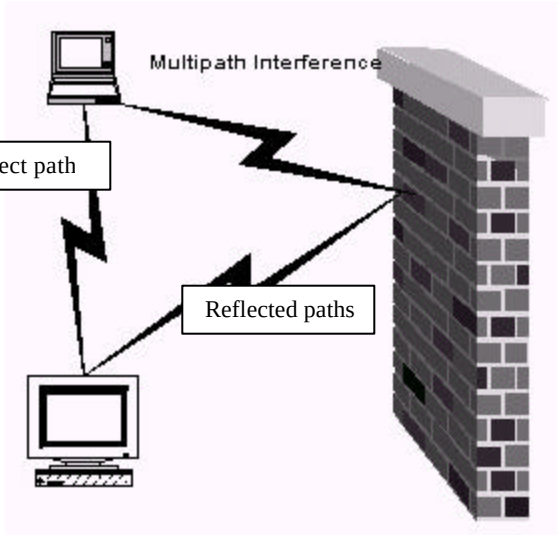


Figure 16. Multipath Interference.

The amount of delay spread varies for indoors home, office, and manufacturing environments, as shown in Table 3. Surfaces of furniture, elevator shafts, walls, factory machinery, and metal constructed buildings all contribute to the amount of delay spread in a given environment [2].

Environment	Delay Spread
Home	< 0.05 μsec (50 nsec)
Office	~ 0.1μsec (100 nsec)
Manufacturing floor	0.2 – 0.3 μsec (200-300 nsec)

Table 3. Typical Multipath Delay Spread For Indoor Environments.

1. Small-Scale Fading And Multipath

Small-scale fading, or simply fading, is used to describe the rapid fluctuation of the amplitude of a radio signal over a short period of time or travel distance. Fading is caused by interference between two or more versions of the transmitted signal which arrive at the receiver at slightly different times. These waves, called *multipath waves*, combine at the receiver antenna to give a resultant signal which can vary widely in amplitude and phase, depending on the distribution of the intensity, relative propagation time of the waves and the bandwidth of the transmitted signal [11].

a. Small-Scale Multipath Propagation

Multipath in the radio channel creates small-scale fading effects. The three most important effects are :

- Rapid changes in signal strength over a small travel distance or time interval.
- Random frequency modulation due to varying Doppler shift on different multipath signals.
- Time dispersion (echoes) caused by multipath propagation delays.

Although a mobile receiver may be stationary, the receiver signal may fade due to movement of surrounding objects in the radio channel. If objects in the radio channel are static, and motion is considered to be only due to that of the mobile, then fading is purely a spatial phenomenon. The spatial variations of the resulting signal are

seen as temporal variations by the receiver as it moves through the multipath field. A receiver moving at high speed can pass through several fades in a small period of time. Maintaining good communication can then become very difficult.

Due to the relative motion between the mobile and the base station, each multipath wave experience an apparent shift in frequency. The shift in received signal frequency due to motion is called the Doppler shift, and is directly proportional to the velocity and direction of motion of the mobile with respect to the direction of arrival of the received multipath wave.

b. Time Dispersion Parameters

In order to compare different multipath channels and to develop some general design guidelines for wireless systems, parameters which grossly quantify the multipath channel are used. The mean excess delay, rms delay spread, and excess delay spread are multipath channel parameters that can be determined from a power delay profile. The time dispersive properties of wide band multipath channels are most commonly quantified by their mean excess delay ($\bar{t}$) and rms delay spread (s_t). The mean excess delay is the first moment of the power delay profile and is defined to be :

$$\bar{t} = \frac{\sum_k a^2{}_k t_k}{\sum_k a^2{}_k} = \frac{\sum_k P(t_k) t_k}{\sum_k P(t_k)} \tag{3-6}$$

The rms delay spread is the square root of the second central moment of the power delay profile and is defined to be :

$$s_t = \sqrt{\bar{t}^2 - (\bar{t})^2} \tag{3-7}$$

Where $\bar{t}^2 = \frac{\sum_k a^2{}_k t^2{}_k}{\sum_k a^2{}_k} = \frac{\sum_k P(t_k) t^2{}_k}{\sum_k P(t_k)}$ (3-8)

These delays are measured relative to the first detectable signal arriving at the receiver at τ_o=0. Typical values of rms delay spread are on the order of microseconds in outdoor mobile radio channels and on the order of nanoseconds in indoor radio channels.

2. Doppler Spread

Delay spread parameters describe the time dispersive nature of the channel in a local area. However, they do not offer information about the time varying nature of the channel caused by either relative motion between the mobile and base station, or by movement of objects in the channel.

Doppler spread is a measure of the spectral broadening caused by the time rate of change of the mobile radio channel and is defined as the range of frequencies over which the received Doppler spectrum is essentially non-zero. When a pure sinusoidal tone of frequency f_c is transmitted, the received signal spectrum, called the Doppler spectrum, will have components in the range f_c -f_d to f_c + f_d, where f_d is the Doppler shift. The amount of spectral broadening depends on f_d which is defined as :

$$f_d = \frac{v_r}{l} = \frac{v_r f_c}{c} \tag{3-9}$$

where c is the speed of light, v_r is the relative velocity, and f_c is the carrier frequency. For example, at 5200 MHz, and a mobile speed of 1m/s (walking speed), the Doppler shift is 17.33 Hz.

3. Types Of Small-Scale Fading

Depending on the relation between the signal parameters (such as bandwidth, symbol period, etc) and the channel parameters (such as rms delay spread and Doppler spread), different transmitted signals will undergo different types of fading. The time dispersion and frequency dispersion mechanisms in a mobile radio channel lead to four possible distinct effects as shown as follows :

Small-Scale Fading (Based on Multipath time delay spread)	
Flat Fading	**Frequency Selective Fading**
1. BW of signal < BW of channel	1. BW of signal > BW of channel
2. Delay spared < Symbol period	2. Delay spread > Symbol period

Small-Scale Fading (Based on Doppler spread)	
Fast Fading	**Slow Fading**
1. High Doppler spread	1. Low Doppler spread
2. Coherence time < Symbol period	2. Coherence time > Symbol period
3. Channel variations faster than baseband signal variations	3. Channel variations slower than baseband signal variations.

4. Fading Effect Due to Multipath Time Delay Spread

Time dispersion due to multipath causes the transmitted signal to undergo either flat or frequency selective fading.

a. Flat Fading

If the mobile radio channel has a constant gain and linear phase response over a bandwidth which is greater than the bandwidth of the transmitted signal, then the received signal will undergo flat fading. In flat fading, the multipath structure of the channel is such that the spectral characteristics of the transmitted signal are preserved at the receiver. However the strength of the received signal changes with time due to fluctuations in the gain of the channel caused by multipath. Figure 17 illustrates how flat fading can distort the amplitude and phase of a received signal. In this scenario a sinusoidal signal is directly transmitted to the receiver and the same signal being reflected and then received. For simplicity, it is assumed that the received signal comprises the sum of the directed signal and the reflected signals. Whether the sum of two such modulated signals cancel or reinforce each other strongly depends on the difference in their phase angles. The phase of the received signal may also differ considerably from the directed signal (see Figure 17(a-f)). If the reflected signal is attenuated, the impact on the amplitude and phase of the received signal becomes limited (Figure 17(b) and (d)). A reflected signal need not always produce a negative effect in a multipath link. As shown in Figure 17(e), although reflected signal 1 virtually cancels out the directed signal, reflected signal 2 actually provides a means to recover the original signal.

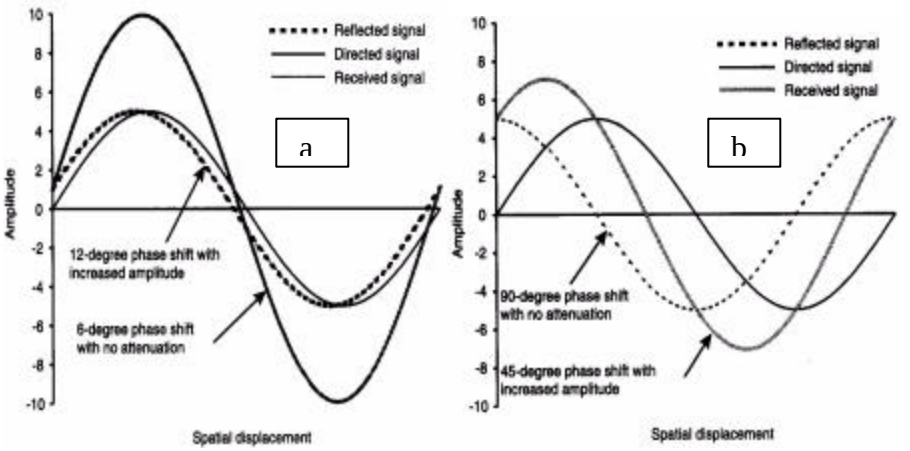


Figure 17(a & b). Impact Of Multipath Reflections On Received Signal.

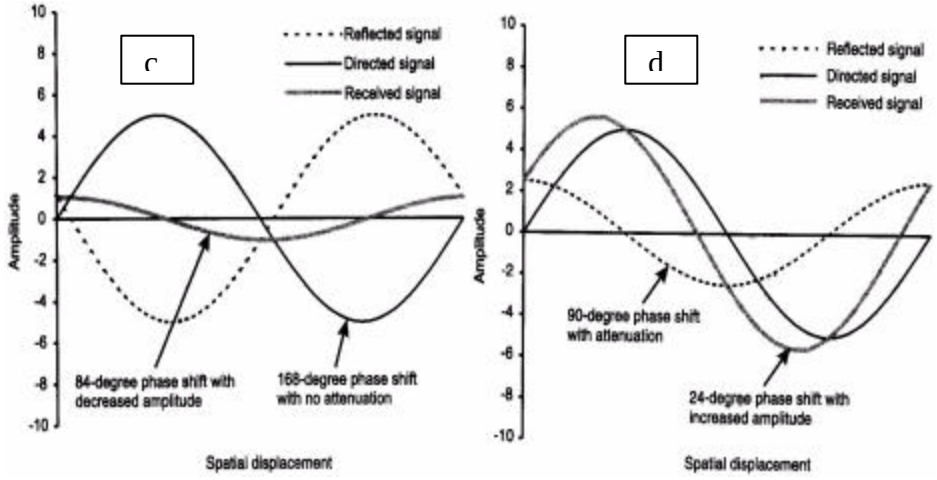


Figure 17(c & d). Impact Of Multipath Reflections On Received Signal.

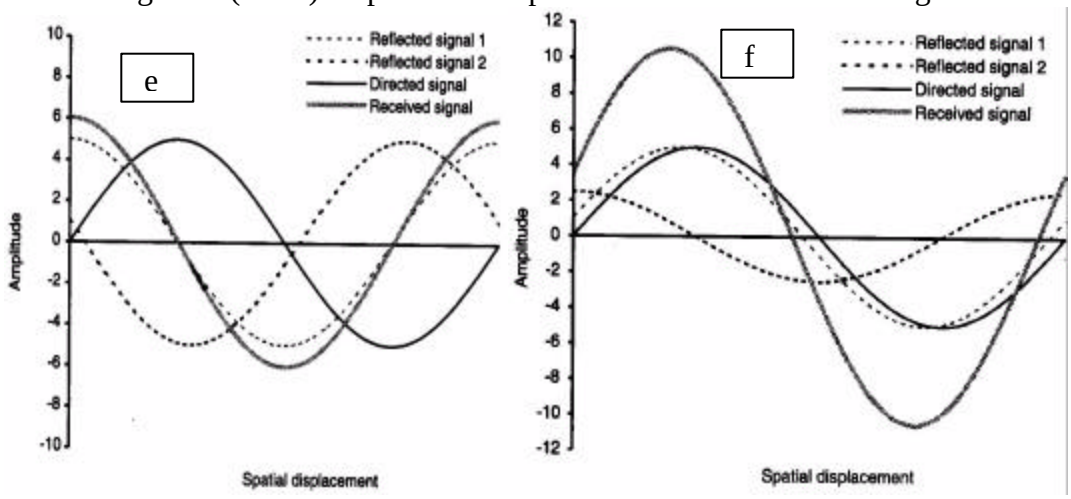


Figure 17(e & f). Impact Of Multipath Reflections On Received Signal.

b. Frequency Selective Fading

If the channel possesses a constant-gain and linear phase response over a bandwidth that is smaller than the bandwidth of transmitted signal, then the channel creates *frequency selective fading* on the received signal. Under such conditions the channel impulse response has a multipath delay spread which is greater than the reciprocal bandwidth of the transmitted message waveform. When this occurs, the received signal includes multiple versions of the transmitted waveform which are attenuated (faded) and delayed in time, and hence the received signal is distorted. Frequency selective fading is due to time dispersion of the transmitted symbols within the channel. Thus the channel induces *intersymbol interference* (ISI).

5. Intersymbol Interference

In general, the effect of the delay spread is to cause the smearing of individual symbols in the case where the symbol rate is sufficiently low, or to further cause time-dispersive fading and intersymbol interference if the symbol rate is high (see Figure 18).

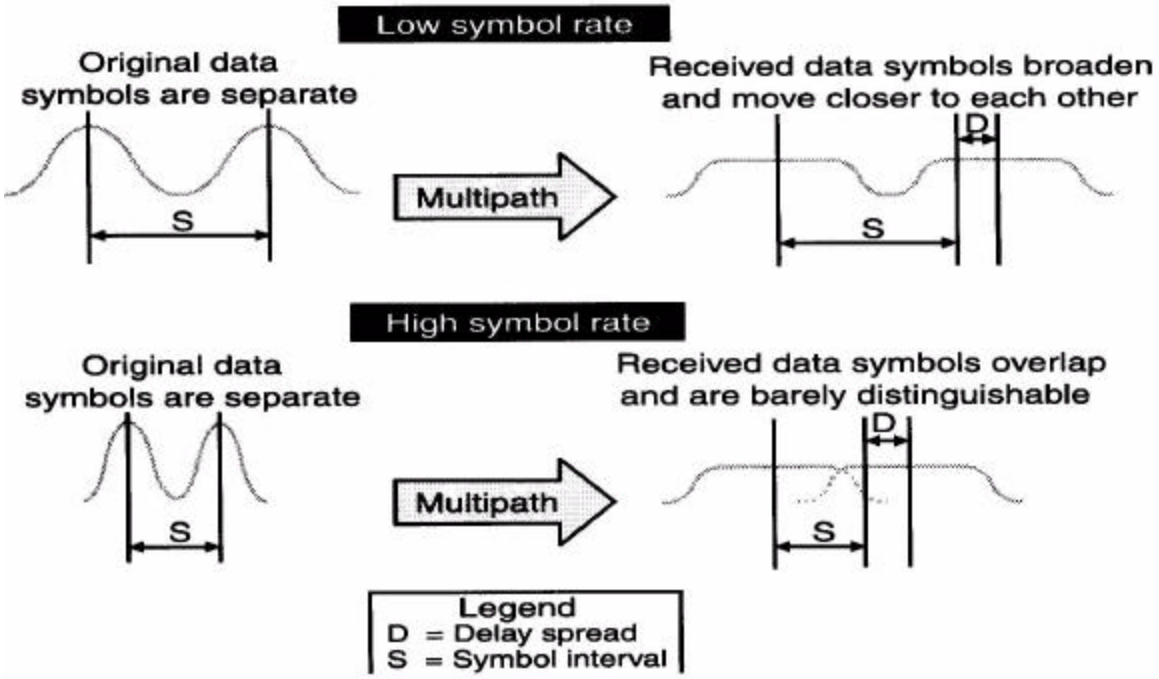


Figure 18. Effects Of Delay Spread.

Intersymbol interference (ISI) is a form of self-interference that increases the error rate in digital transmission, an impairment that cannot be overcome simply by improving the signal-to-noise ratio. This is because increasing the signal power in turn increases the self-interference. At higher symbol rates or larger delay spreads, the difference in delay among the various signal reflections arriving at the receiver can be a significant fraction of the symbol interval. Normally, a delay spread of more than half a symbol interval results in indistinguishable symbols and a sharp rise in the error rate.

The ISI is prevented in IEEE 802.11a (with COFDM) by creating a cyclically extended guard interval, where each OFDM symbol is preceded by a periodic extension of the signal itself.

6. Path Loss

In general, the spatially averaged power P_o at a point a distance d from the transmitter is a decreasing function of d. Usually, this function is represented by a path-loss-power law of the form :

$$P_o \approx d^{-g} \tag{3-10}$$

In free space the path-loss law exponent g =2, so the power law obeys an inverse-square law [12]. For WLAN, the signal attenuation is dependent not only on distance and transmitted power but also on reflecting objects, physical obstructions, and the amount of mutual interference from other transmitting nodes. While the free-space exponent may be relevant for short distance transmition (eg. up to 10m), the path loss is usually modeled with a higher-valued exponent of 3 to 5 for longer distances [18].

7. Rayleigh And Ricean Distribution

a. Rayleigh Fading Distribution

A Rayleigh distribution is commonly used to describe the statistical time varying nature of the received envelope of a flat fading signal, or the envelope of an individual multipath component.

b. Ricean Fading Distribution

When there is a dominant stationary (nonfading) signal component present, such as a line-of-sight propagation path, the small-scale fading envelope distribution is Ricean. In such a situation, random multipath components arriving at different angles are superimposed on a stationary dominant signal

The effect of a dominant signal arriving with many weaker multipath signals gives rise to the Ricean distribution. As the dominant signal becomes weaker, the composite signal resembles a noise signal which has an envelope that is Rayleigh. Thus, the Ricean distribution degenerates to a Rayleigh distribution when the dominant component fades away.

D. IEEE 802.11 CHANNEL MODEL

In an environment where performance measurement of the same radio is used in the same location, over time the results may not agree. This is due to the changing position of people in the room and slight changes in the environment. These can produce significant changes in the signal power at the radio receiver. A consistent channel model is required to allow comparison of different WLAN systems and to provide consistent results. In doing so, the IEEE 802.11 Working Group adopted the following model as the baseline for predicting multipath in modulations schemes used in IEEE 802.11a and IEEE 802.11b. This model is ideal for software simulations prediction performance results of a given implementation. The channel impulse response illustrated in Figure 19

is composed of complex samples with random uniformly distributed phase and Rayleigh distributed magnitude with average power decaying exponentially [2].

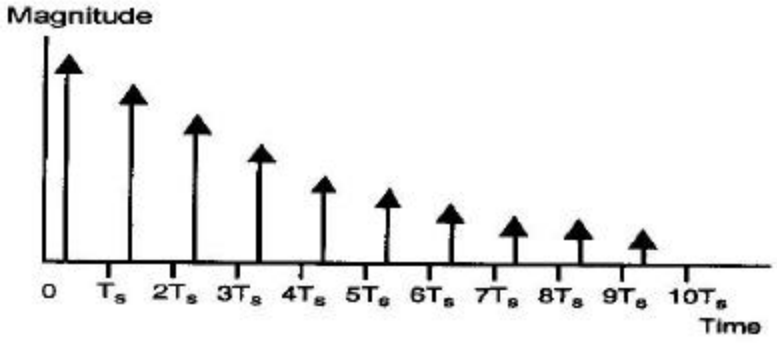
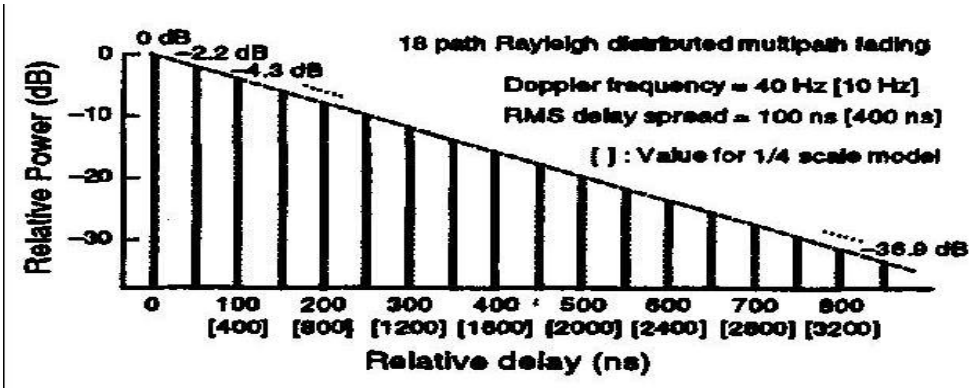


Figure 19. Channel Impulse Response For IEEE 802.11a.

Similar models were implemented in [10] and [15], and the multipath delay profile is shown in Figure 20. It consists of 18 path signals at interval of 50ns. This model is chosen and implemented in this thesis.



Delay Profile (in nsec)	Gain Profile (in dB)	Delay Profile (in nsec)	Gain Profile (in dB)
0	0	450	-19.54
50	-2.17	500	-21.71
100	-4.34	550	-23.89
150	-6.51	600	-26.06
200	-8.69	650	-28.23
250	-10.86	700	-30.40
300	-13.03	750	-32.57
350	-15.20	800	-34.74
400	-17.37	850	-36.92

Figure 20. Multipath Delay And Gain Profile.

In this chapter we first discussed the AWGN noise channels. The AWGN noise has no multiplicative mechanisms, and is noise that is simply superimposed or added to the signal. It is implemented in this thesis so as to investigate its effect on the performance of coded OFDM. We have covered multipath which is one of the performance concerns for indoor IEEE 802.11 WLAN systems. A consistent multipath channel model adopted by the IEEE 802.11 Working Group, and implemented in thesis is also discussed in this chapter.

IV. MATLAB COFDM SYSTEM MODEL

A. GENERAL

The next step in the research was the implementation of a COFDM computer system model. The work of [17] was adopted for the purpose of this thesis. For this thesis, all signal processing and channel transmission through the simulated links are performed at baseband. Since the objective of this thesis is to emulate and simulate the physical layer, it is deemed not necessary to have a physical implementation, hence, the functions normally associated with RF up-conversion and down-conversion are not necessary to generate meaningful tradeoff results. Thus, filtering, digital-to-analog conversion (DAC), up/down frequency translation and analog-to-digital conversion (ADC) functional sub-blocks necessary for actual implementation are not included in the computer model. A block diagram of the complete system model which is emulated in MATLAB and simulations performed is presented in Figure 21.

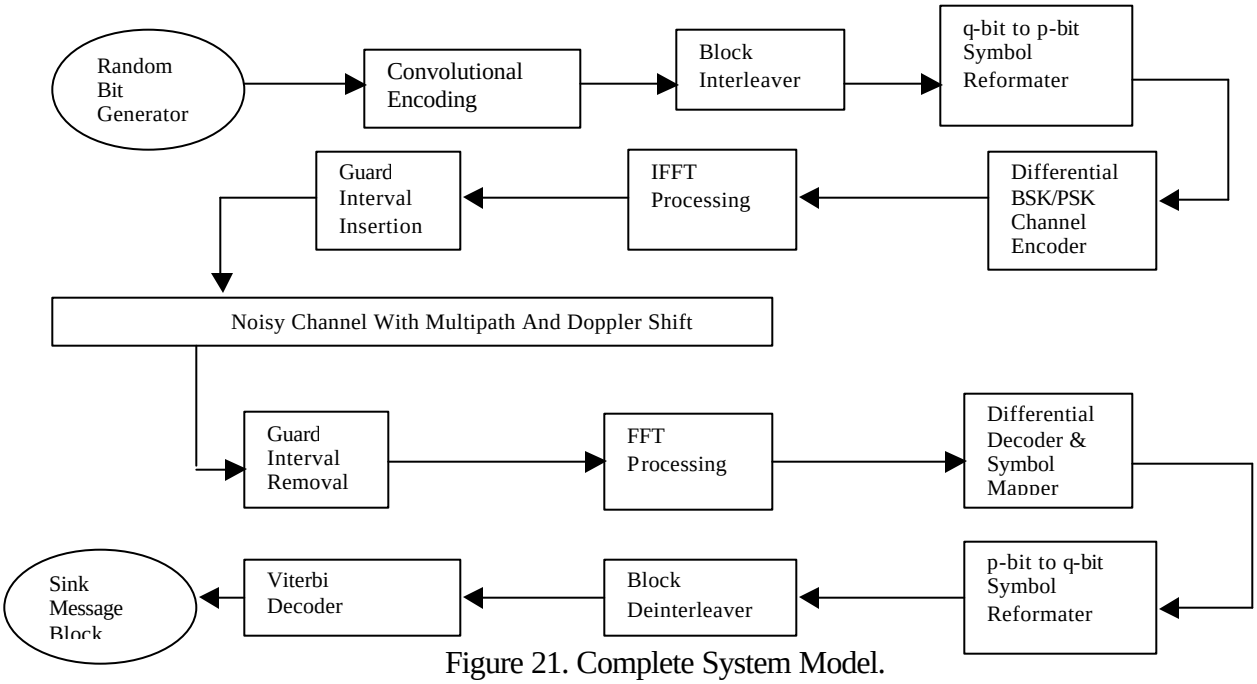


Figure 21. Complete System Model.

B. COFDM TRANSMITTER

The COFDM transmitter functional block diagram is illustrated in Figure 22 with each of the sub-blocks subsequently described.

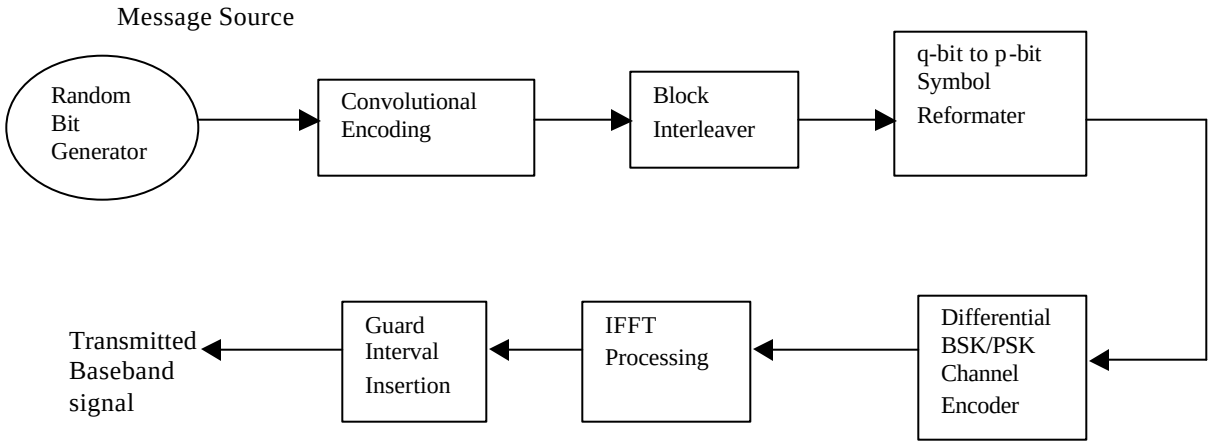


Figure 22. COFDM Transmitter Functional Block Diagram.

1. Random Bit Generator

This functional block originates a random message bit pattern representing the information source. The bit sequence length is variable as defined by the user. The random property of each binary element is determined by a seed parameter setting the internal computer's random number generator seed. If multiple simulations are performed using the same seed values, identical results occur. This property is useful when comparing and contrasting simulation outputs with different system configurations. By fixing seed values, optimal system configurations can be ascertained based upon superior BER performance while using consistent channel characteristics and source message symbol patterns. It is also possible to set the seed randomly by the internal PC processor.

2. Convolutional Encoding

Convolutional coding is a special case of error-control coding. A convolutional coder is not a memoryless device. Even though a convolutional coder accepts a fixed number of message symbols and produces a fixed number of code symbols, its computations depend not only on the current set of input symbols but on some of the previous input symbols.

3. Block Interleaver

A block interleaver accepts a set of symbols and rearranges them, without repeating or omitting any of the symbols in the set. The number of symbols in each set is fixed for a given interleaver. The interleaver's operation on a set of symbols is independent of its operation on all other sets of symbols.

4. Symbol Reformatter

In preparation for the appropriate N-ary modulation scheme, $N=2^P$ (Note : the N used for N-ary signaling is not the same N used for N-point FFT calciulations). Since 2-PSK (BPSK) and 4-PSK (QPSK) are predominately used during simulation runs, symbol lengths are resized as either 1-bit (p=1) or 2-bit (P=2) length words. If necessary, zero bit padding may be required during the reformatting process to account for incomplete word formations.

As a result of symbol reformatting, the dimensions of the original source message array may change to compensate for the addition or deletion of redefined symbols. Regardless of the number of new PSK symbols formed, the number of matrix columns corresponding to OFDM subcarriers remains fixed. Hence, any necessary message symbol quantity adjustment is accommodated by increasing or decreasing the number of matrix symbol rows instead. For example, if during the symbol reformatting process the OFDM symbols are changed from 8-bits to 4-bits, then the total number of message symbols double from their original amount. Consequently after reformatting, the number of message matrix rows double while the number of message matrix columns remains constant.

5. Differential PSK Channel Encoder

PSK is the preferred modulation technique for channel encoding in multipath channels. Prior to signal constellation mapping, differential encoding is performed on the symbols within the message matrix. Two types of differential encoding are included.

Considering differential encoding along the time dimension (symbol rows), a cumulative summation down each column of the message symbol array is calculated. For differential encoding along the frequency dimension (OFDM frequencies), a cumulative summation across each row of the message symbol array is calculated. Recall that construction of the message block matrix is designed so that column represent OFDM frequencies (frequency dimension), while rows represent symbols generated in time (time dimension). During subsequent simulation trials, either frequency and/or time differential encoding may be selected to evaluate system performance.

The differential encoding/deconding technique introduces memory into the system and allows for decoding of the current received symbol with respect to the previously decoded symbol. Consequently, detection decisions are based upon relative differences between consecutively received symbols. This technique may be advantageous in a slowly fading multipath channel where the variations among successive received symbols are negligible. A cumulative summation can be best illustrated through an example.

Given V = [1 2 3 4 5 6 7 8 9]' (4-1)

Then, $CsumV_{16}$ = [1 3 6 10 15 5 12 4 13]' (4-2)

V is a column vector whose elements represent message symbols taken from the set of N integers, where $N=2^P$. CsumV is formed by consecutively adding in modulo-N fashion successive elements in V beginning with one to the current running total in CsumV beginning with zero. For this example N=16; thus, 0 +1 =1, 1+2 (the next element in V)=3, 3 + 3 =6, 6+4=10, 10+5=15, 15+6=21 =5 (modulo-16) and so on. In this way, all the elements in CsumV are calculated with respect to the first element in V. A more concise expression is :

$$C_{sum}V_k = V_k \oplus C_{sum}V_{k-1} \quad (4\text{-}3)$$

where $\{V_k\}$ is a modulo-N message sequence input to the differential encoder, $\{C_{sum}V_k\}$ is the encoder output sequence, and $\oplus$ denotes modulo-N addition.

Following differential encoding, each symbol in the differentially encoded message array is channel encoded as a complex modulation value with unit magnitude and one of N possible phases (N-PSK modulation) ; that is,

$$D_k = e^{\frac{2pjCsumV_k}{N}} \qquad (4\text{-}4)$$

In continuation of the previous example (i.e. $CsumV_{16}$ = [1 3 6 10 15 5 12 4 13]'), the corresponding vector of 16-ary complex modulation value phase angles are,

$$Ang[D]=[22.5^o\ 67.5^o\ 135^o\ 225^o\ 337.5^o\ 112.5^o\ 270^o\ 90^o\ 292.5^o]' \qquad (4\text{-}5)$$

A row of ones representing zero phase complex modulation values is appended to the top of the message array during time differential encoding, representing a decoding reference for the receiver. For frequency differential encoding, a pair of columns containing ones elements is appended to the extreme left side of the message array as a similar decoding reference. Two ones columns are included instead of a single column to maintain an even number of OFDM frequencies (even number of columns).

6. IFFT Processing

To convert the frequency array to time domain representation, an N-point IFFT is performed producing a corresponding output sequence of time domain samples. The input array complex modulation values have the left and right half swapped by the previous frequency arranger block to account for the automatic frequency index shift that results from the IFFT.

7. Guard Interval Insertion

A guard interval composed of a period extension of the symbol is inserted at the beginning of each symbol for channel impluse response compensation purposes. The length of the guard interval is fixed at 800ns to account for multipath delays. The guard interval is represented by additional 16 time domain samples added to the resulting sequence derived from IFFT processing.

C. COFDM RECEIVER

The receiver functional block diagram is illustrated in Figure 23. The blocks in the receiver perform the reciprocal functions of the transmitter and are described as follows.

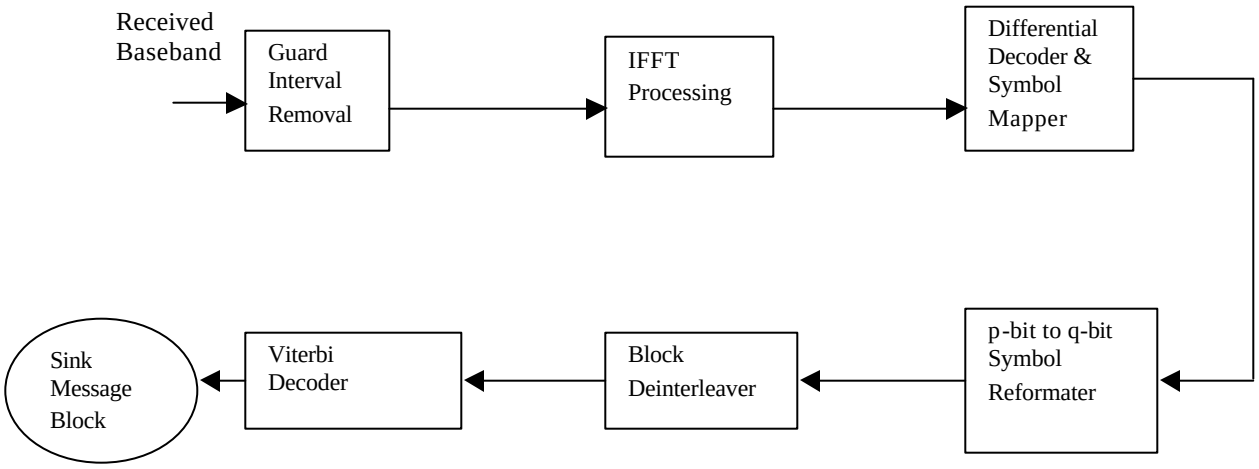


Figure 23. Receiver Functional Block Diagram.

1. Guard Interval Removal

The guard interval precursor appended to each symbol in the transmitter is initially removed, leaving behind the remaining information portion of the symbol for further processing. The information symbol consists of a sequence of 16 time domain samples.

2. FFT Processing

The sequence of time domain samples are transformed into the frequency domain using an 64-point FFT to recover the OFDM frequency tones information. In a linear time-invariant channel, the orthogonality of carriers is preserved; however, in a multipath environment with frequency Doppler shifting, this is not always the case. The output is an array of complex modulation values with the left half portion shifted to the right N positions as a result of the FFT operation.

3. Channel Decoding

Differential decoding is first performed either in the frequency dimension (matrix column) or time dimension (matrix rows), maintaining compatibility with the transmitter differential encoding method. In addition, the previously appended reference ones elements are removed. Afterwards, channel decoding is accomplished, inverse mapping each received complex modulation value with magnitude and phase into a corresponding N-ary symbol representation composed of p bits. Considering QPSK, 2 bit long symbols are reconstructed.

4. Block Deinterleaving

The message is next deinterleaved to reconstruct proper ordering of the information symbol stream according to the particular interleaving configured in the transmitter. After deinterleaving, any corrupted symbol errors caused by burst noise in the channel should be sufficiently redistributed within the message array, creating a more random, uncorrelated error distribution.

5. Received Message

The output of the receiver represents the received sink message block. After transmission through the system channel model prone to noise and multipath distortions, symbol errors may exist. The distribution of error events within a message array is recorded and the bit error rates calculated to generate corresponding performance curves. The resulting simulation data is compared to the theoretical performance criteria for evaluation.

D. CHANNEL MODELS

Three channel models are emulated as part of the overall communication system model and used during simulations (a noise free channel 0 model is also included for system functional verification) (Figure 24). One emulated channel type is the AWGN

model and represents additive noise only. The second is the multipath channel model and is characterized by frequency selective fading (loss) in dB, Doppler frequency shifting in Hz and multipath time delays in microseconds which vary for each transmission link according to the specified multipaths. The composite channel 3 model is a combination of channel 1 and channel 2 models; thus, the AWGN model is added to the multipath model representing the actual communication environment.

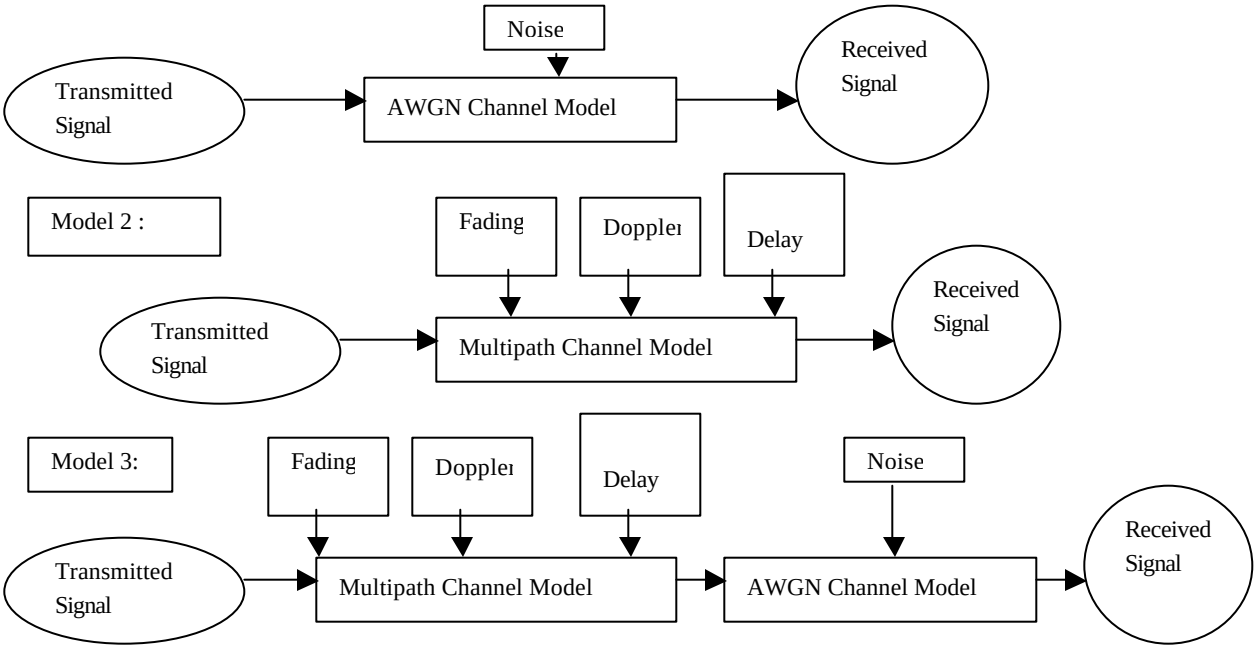


Figure 24. Channel Models.

V. MATLAB PROGRAMMING AND DEVELOPMENT

A. OFDM SYSTEM CONSTRUCTION OF FUNCTIONAL BLOCKS

Emulation of the COFDM communication system as shown in Figure 25 is done by initially portioning the overall system according to functionality and forming functional interconnecting subblocks. The COFDM system model consists of three primary components : a COFDM transmitter, the channel and a COFDM receiver. Within the transmitter are two separate functional blocks, a source encoder block and an IFFT processing block. The channel consists of four separate models: the channel 0 model, the channel 1 model, the channel 2 model and the channel 3 model. Each channel model corresponds to a different type of noise (except for the channel 0 model which is noise free). The receiver block consists of two blocks : the FFT processing block and the message decoding block. Recall that all simulations are preformed at baseband; therefore, no additional block associated with RF bandpass transmissions are required nor included in the model.

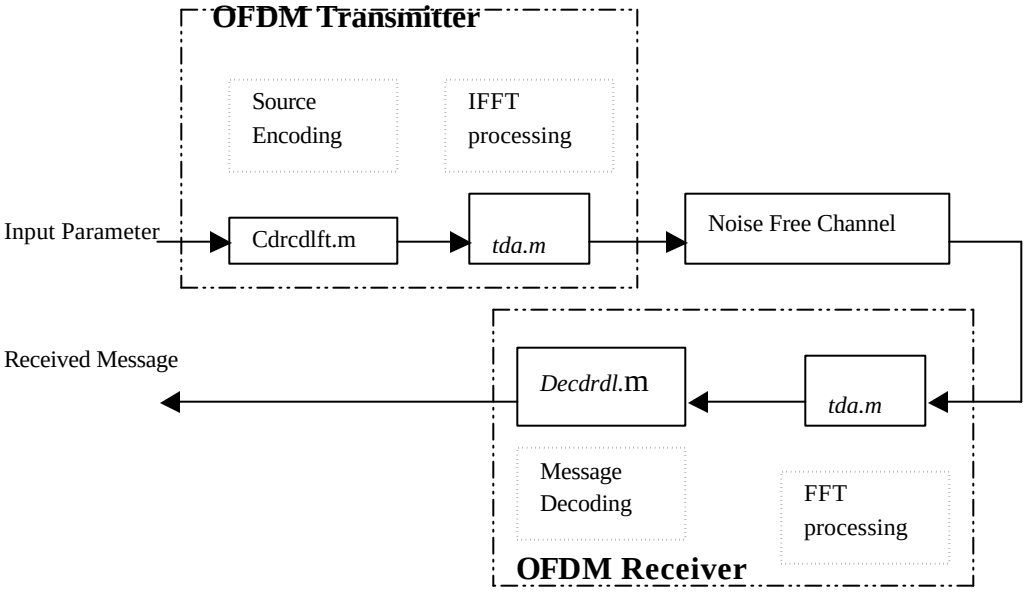


Figure 25. Emulation Of The COFDM Communication System.

The model 0 block diagram is shown in Figure 26 and represents a noise free perfect channel. (i.e. , the absence of AWGN and any multipath influence within the channel). Transmitter source encoding is performed within the m-file macro, ***cdrcdlft.m***. The functional sub-blocks associated with ***cdrcdlft.m*** are depicted in Figure 27. The IFFT processing block responsible for generating OFDM frequency tones and appending guard intervals is represented by the m-file macro, ***tda.m***. Correspondingly in the receiver, the inverse functions of the transmitter are performed, namely FFT processing and guard interval removal is accomplished by the ***itda.m*** m-file, while signal decoding is accomplished by macro ***decdrcdl.m.***

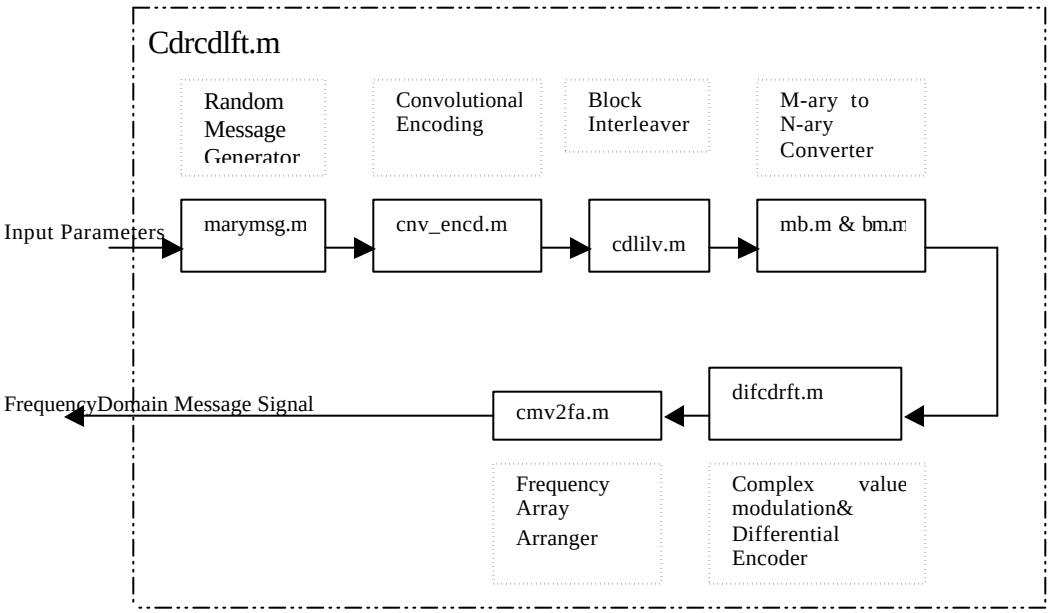


Figure 26. Model 0 Block Diagram.

B. COFDM TRANSMITTER

The hierarchical arrangement of m-files within ***cdrcdlft.m***, including subroutine macros, are presented in Figure 27 and are subsequently described in detail.

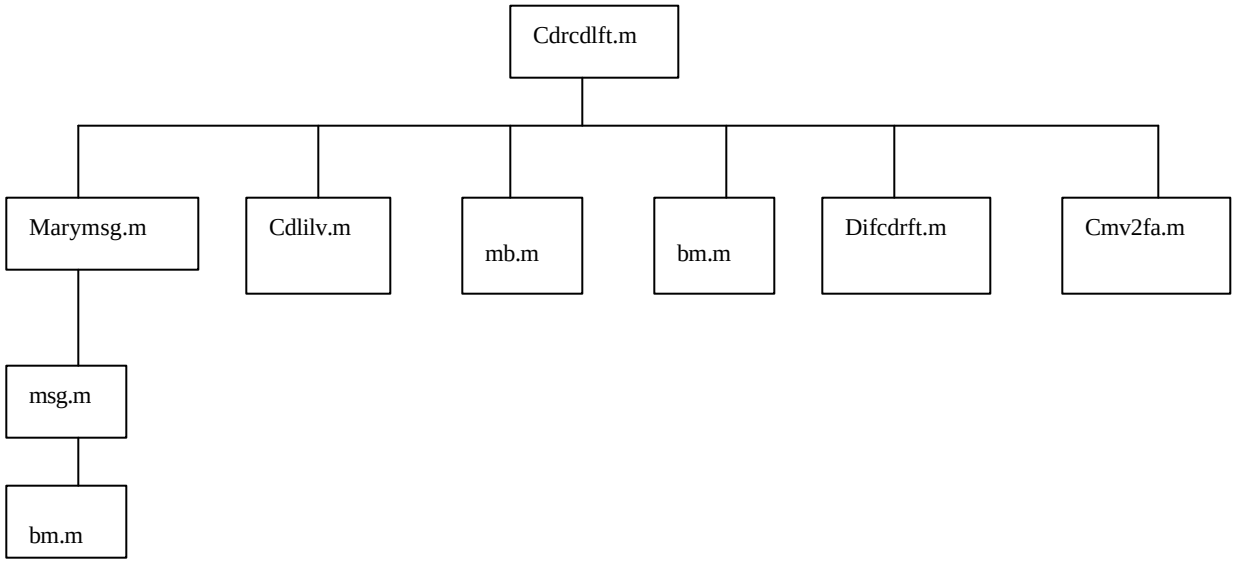
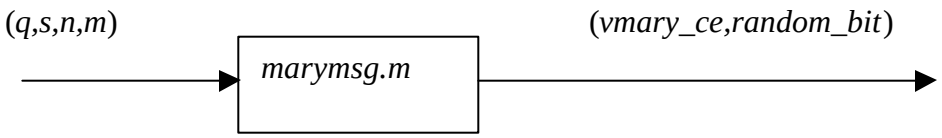


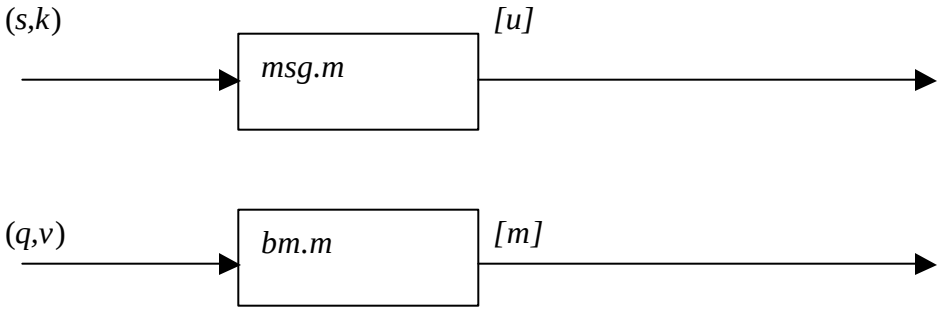
Figure 27. Hierarchical Arrangement Of M-files Within Cdrcdlft.m.

The source message is randomly generated by the m-file ***marymsg.m***. The general form of the function is depicted by the functional block shown below.

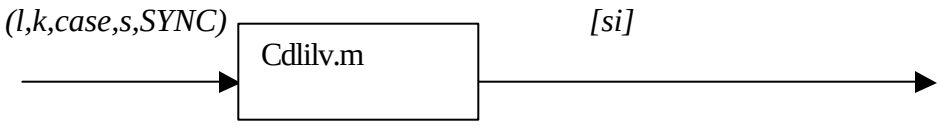


This function first generates an array of randomly generated ***q***-bit long symbols representing the random bit source, ***random_bit***. The random_bit source is then fed into a convoultional encoder which generates the coded message of ***vmary_ce***. The input arguments, ***n*** and ***m***, determine the overall output message matrix dimensions, where ***n*** is the number of rows and ***m*** is the number of columns. The value selected for ***m*** also represents the number of OFDM frequency tones and must be an even positive integer so as to completely fill the available transmission bandwidth without spectral cutoff of the endpoint symbols. The value selected for ***n*** is any arbitrary positive integer and represents rows of symbols generated in time. The input argument, ***s***, is the seed

parameter used for setting the seed of the internal MATLAB random number generator function. The remaining input argument, ***q***, represents the number of bits contained in each of the symbol words considering M-ary signaling, $M=2^q$. The function ***marymsg.m*** requires three other subroutine m-files, ***msg.m, bm.m*** and ***cnv_encd.***



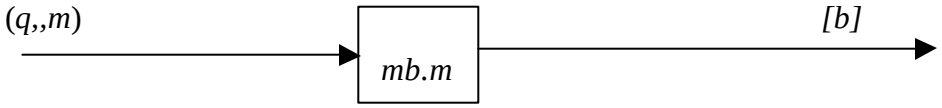
The function ***msg.m*** randomly generates a k-length binary output sequence, ***u***, with the random number generator seed set by parameter, ***s.*** The function ***bm.m***, representing a binary to M-ary converter, transforms a variable length binary input sequence, ***v***, into an equivalent M-ary output sequence, ***m***, depending on the value selected for ***q***, the word bit length. By accepting as an input the random binary output generated by m-file ***msg.m***, ***bm.m*** groups bits together ***q***-bits at a time to form words representing M-ary symbols whose output is a vector of equivalent decimal numbers. Padding with zeros may be necessary to ensure a complete ***q***-bit word formation.



After the randomly generated source message is encoded by the convolutional encoder, the array is next interleaved by the m-file function ***cdlilv.m***. This m-file has a five argument input and a single output. Parameters, ***l*** and ***k***, determine the dimensions of

the interleaver intermediate matrix where ***l*** is the number of rows and ***k*** is the number of columns. The parameter, ***case***, is an input that selects which desired interleaveing method should be included. There are nine different interleaving cases. Case 0 represents a conventional block interleaver which is used for simulation in this thesis. Case 1 through 8 are not necessary and therefore are not used.

After the interleaving operation, the interleaved message array is converted from an M-ary format to a N-ary format suitable for N-PSK modulation. The symbol format conversion process is accomplished by two separate m-file routines, ***mb.m*** and ***bm.m***. The function ***mb.m*** accepts two input variables and represents a M-ary to binary converter. The input ***q*** is the number of bits defining the M-ary symbols where $M=2^q$. The remaining input, ***m*** represents the incoming M-ary message array. The single output from this block, ***b,*** is a binary data sequence whose information content is equivalent to the coded M-ary symbols.

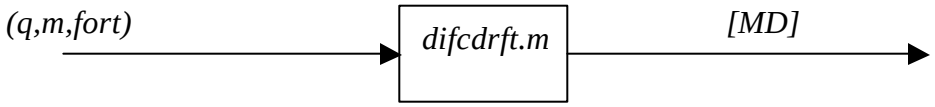


The binary output sequence generated by ***mb.m*** is next fed as an input to ***bm.m***. Recall that the function ***bm.m*** converts a variable length binary input sequence, ***v***, into an equivalent N-ary output symbol sequence, ***m***, where $N=2^q$. In this way, the combination of m-files ***mb.m*** and ***bm.m*** functions effectively convert the interleaved message information block from an array containing M-ary symbol to one consisting of N-ary symbols.

With the desired bit values determining M and N chosen by the user, the size of the N-ary message array may change since additional symbols may be formed, or likewise there may be a reduction in the number of symbols. However, the number of columns in the final symbol message matrix consistently remains unaltered as they represent the number of OFDM sub-carriers and remain fixed for each simulation. If the message block size must increase or decrease as a result of M-ary to N-ary symbol format

conversion, the adjustment is accomplished by increasing or decreasing the number of rows in the message block only, never the number of columns. For example, given an arbitrary message array to be converted form M-ary symbol format to equivalent N-ary symbols, the input message array will increase two-fold from the original total. Consequently, the number of rows forming the output matrix doubles, while the column number remains the same. As a function of the desired M-ary and N-ary configuration, a pad of zero symbols may be automatically inserted to ensure a full array. In the receiver, the zero pad is removed, leaving behind the randomly generated message source.

After the interleaving and M-ary to N-ary conversion operations are accomplished, the message array containing information symbols represented in decimal notation, is differentially encoded then channel encoded as an array of complex modulation values suitable for N-PSK modulation. The symbol-to-complex-modulation-value mapping process is accomplished using the m-file, ***difcdrft.m***. This function has a three argument input and a single output consisting of differentially encoded complex modulation values, ***MD***, in array format.



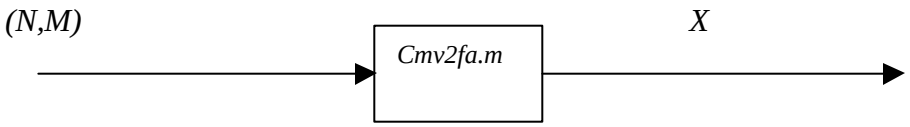
The input, ***fort,*** determines how the array, ***m***, is processed. If ***fort*** is zero, time differential encoding is performed on the message array, ***m***, by executing a cumulative summation down each column. If ***fort*** is one, frequency differential encoding is performed by similarly performing a cumulative summation across each row in the message array, ***m***. Recall that array columns correspond to OFDM frequencies, while array rows represent information symbols generated in time.

Cumulative summations of the input array are accomplished by adding in modulo-N fashion the first element of the appropriate column or row vector to the next adjacent element, replacing the second element by the current summation, then adding this current sum to the third element and replacing that element with the current sum. This process is repeated until all elements in the row (frequency differential encoding) or

column (time differential encoding) are exhausted. The cumulative summation process is then repeated beginning with the first element of the next row of column respectively.

After differential encoding with modulo-N cumulative summations, the array, ***m***, is channel encoded as N-ary complex modulation values. The input, ***p***, indicates the number of unit circle phase partitions formed based upon the N-PSK modulation scheme where $N=2^p$. The mapping process begins by accepting the input symbol message array, ***m***, and generating corresponding complex modulation values, ***MD***, with unit magnitude and one of N possible phases. Recall that complex modulation numbers are described by a magnitude of one(A=1) and possible phase values selected from the set, {± 22.5,±45,±67.5,±90,±112.5,±135,±157.5,0,180} degrees.

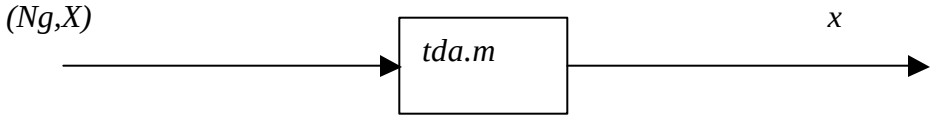
As a final step, a reference row of ones (zero phase angles) are appended to the message array, ***m***, at the top to provide a reference starting point for the differential decoding performed in the COFDM receiver. Similarly, for frequency differential encoding, a reference column pair of ones (zero phase angles) are appended to the message array, ***m*** at the left. Two reference ones columns are appended to maintain an even number of OFDM frequencies. Consequently, ***MD*** includes the additional reference ones within the complex modulation array. In the receiver, these reference values are stripped off during differential decoding.



As a final step in the source encoding block and in preparation of OFDM frequency generation through the IFFT, the input array of complex modulation values, M, are rearranged into a special frequency array by the m-file ***cmv2fa.m***. The second input variable, N, is the number of FFT points used which must be larger than the number of columns of complex modulation values in the array (number of OFDM frequencies). This function also swaps the positions of the modulation values by grouping the left half portion of the matrix elements and shifting them to the rightmost positions, and likewise

grouping the right half portion of the matrix elements and shifting them to the leftmost positions. Swapping is performed in anticipation of the frequency spectrum shifting that automatically results from FFT processing. When the MATLAB FFT command is invoked, the negative spectral frequencies are shifted to the rightmost positive locations by N positions. Thus, the spectrum is no longer symmetrical about the origin but instead becomes symmetrical about the frequency point N/2. If the frequency halves are swapped before IFFT processing, then the frequencies can be later recovered in their correct orientation by filtering.

The shifted frequency array output is represented by ***X***. A pad of zeros is included in the middle of the array whose amount is the difference between the number of FFT points, N, and the number of modulation values. The zero pad is included as a guard band to account for filter slopes during subsequent bandpass filtering after up-conversion and RF transmission. This filtering is not actually performed for the thesis simulations, however, the guard band is included for actual implementation purposes.



After source encoding, the complex modulation frequency array, ***X***, is IFFT processed within the m-file, ***tda.m***, generating the OFDM frequencies. The ***tda.m*** function also prepares the transmitted symbols for channel compensation by first appending the periodic guard interval whose length is indicated by the input, ***Ng***. ***Ng*** represents the number of additional time domain waveform samples to add to the beginning of the information symbol interval. The output, ***x***, is the time domain samples suitable for transmission and consisting of an array of complex samples. This functional block is the final block the message signal enters before transmission through the channel. Again, for purposes of this thesis, DAC and up-conversion of the signal is not included, permitting all simulations to be performed at baseband.

C. COFDM RECEIVER

1. Model 1 System

The model 1 block diagram is shown in Figure 28 and represents the channel 1 model consisting of the AWGN channel, implemented using the m-file ***awgn.m***.

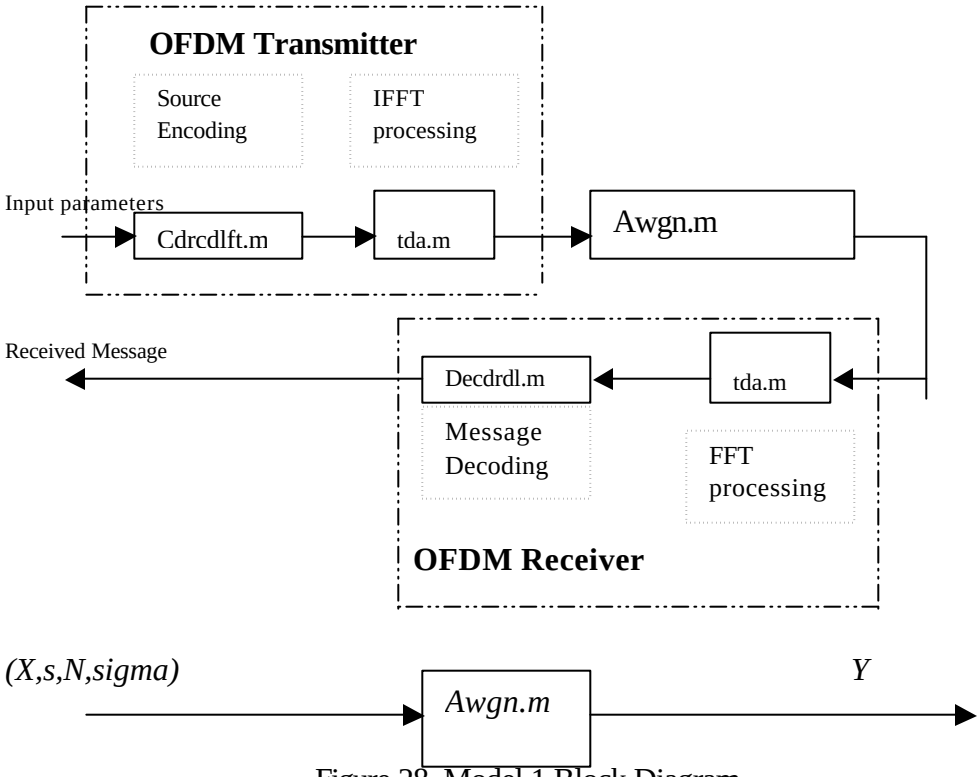


Figure 28. Model 1 Block Diagram.

The receiver decoding functions are performed within the ***decdrcdl.m*** block by multiple sub-blocks which are presented above in Figure 28. The hierarchical arrangement of m-files within decdrcdl.m are presented in Figure 29.

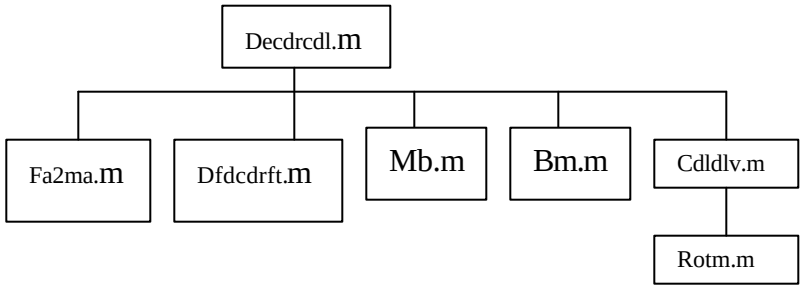
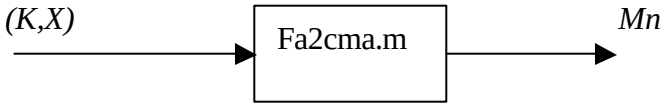
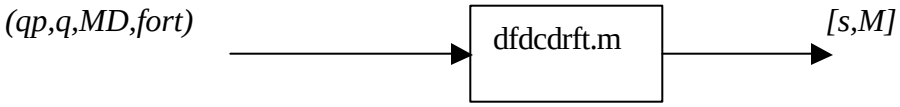


Figure 29. M-file Hierarchy for Decdrcdl.m.

The frequency array is restructured back into the proper complex modulation array format by the ***fa2cma.m*** m-file within ***decdrcdl.m***. The function ***fa2cma.m*** accepts the input ***K*** indicating half the number of OFDM frequency tones (corresponds to frequencies occupying one-half of the frequency array). The remaining input, ***X***, are the complex frequency array values to be rearranged. The output, ***Mn***, is the equivalent complex modulation array representation with the correct ordering of frequencies.



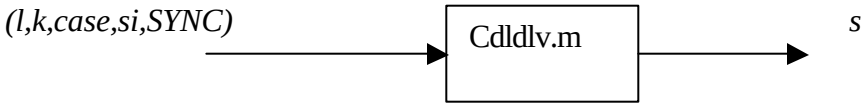
After the ***fa2cma.m*** block, the complex modulation values are differentially decoded either in time or in frequency, then decoded into corresponding N-ary symbols.



This functionality is accomplished by the m-file ***dfdcdrft.m.*** The complex modulation values, ***MD***, from ***fa2cma.m*** are accepted as an input, and inverse mapping of the complex numbers to N-ary symbols is performed based upon the value of ***q***, where $N=2^q$. If ***fort*** is equal to one, frequency differential decoding is performed. Differential decoding is the inverse operation performed in the transmitter; however, regardless of the type of differential decoding, all reference one values are removed after decoding allowing the received message matrix to remain. The output, ***s*** indicates phase sector numbers corresponding to N-ary demodulation also representing corresponding inverse mapped symbols in decimal notation. The remaining output, ***M***, is the differentially decoded modulation array.

With the reception of the message in N-ary format consisting of PSK symbols, a reformatting of symbols to M-ary is next performed to form OFDM symbols. Once again

the functions ***mb.m*** and ***bm.m*** perform the reformatting procedure as previously described in the transmitter section.



As a final operation in the receiver, the message symbol array is deinterleaved by the function ***cdldlv.m*** which performs the inverse operation of ***cdlilv.m***. The input, ***si***, is the received interleaved message, while case determines which deinterleaving ***case*** to follow (block interleaving is used for this thesis). The output, s, provides the final message array read out of the intermediate matrix by rows. ***Cdldlv.m*** calls the subroutine m-file, ***rotm.m*** which performs the array rotations as previously described in ***cdlilv.m***.

2. Model 2 System

The COFDM model 2 system is presented in Figure 30 and has identical transmitter and receiver components as the model 1 system, differing only in the channel model. The channel 2 model consists of the multipath channel exclusively which is implemented using the ***chuhf.m*** m-file. No other types of noise such as AWGN are added to this model; thus, the multipath effects on the transmitted signal can be individually analyzed.

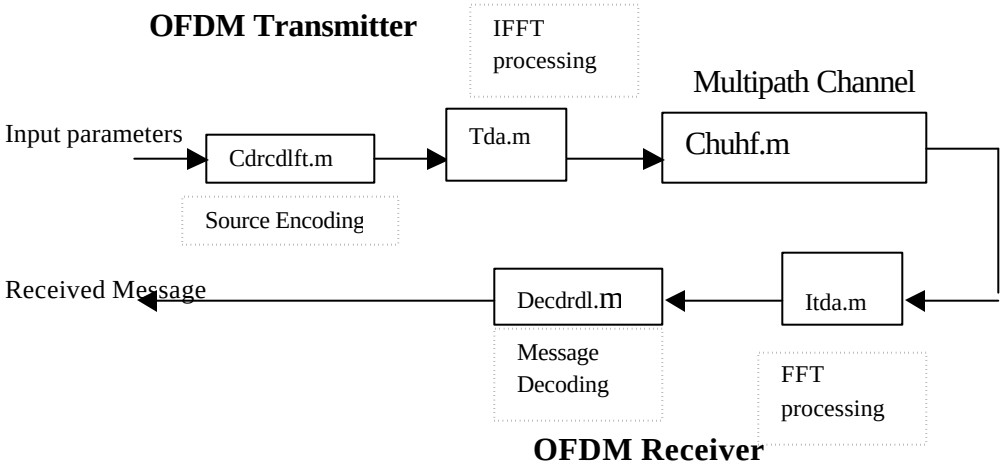


Figure 30. System Model 2 Block Diagram.

The m-file ***chuhf.m*** represents the channel 2 multipath model. The hierarchy for ***chuhf.m*** is shown in Figure 31.

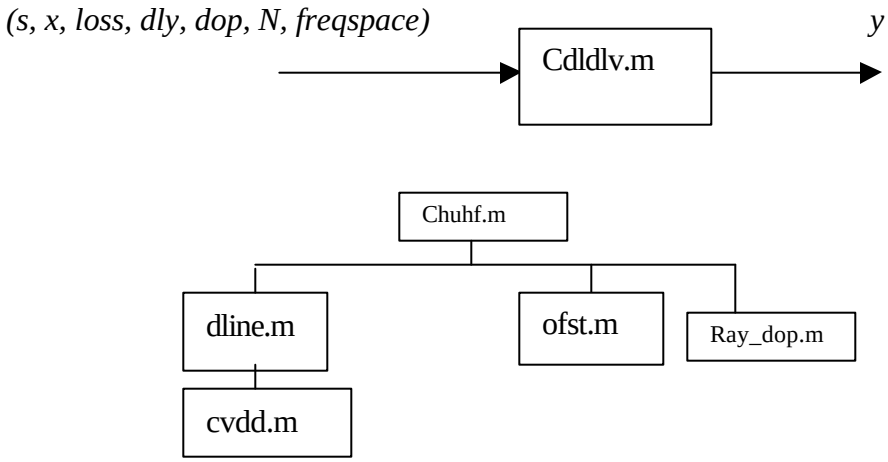


Figure 31. M-file Hierarchy For Chuhf.m.

This m-file accepts as inputs the Received Signal Loss, ***loss*** (dB), time delays, ***dly***, and Doppler frequency shifting, ***dop*** (Hz). The transmitted signal, ***x***, represents the time domain output of the COFDM transmitter consisting of complex numbers and is the input signal parameter to the channel model. Initially, the m-file ***dline.m*** is called to set-up the multipath delayed paths. Since the input, ***dly***, can be a vector of delays, the number of delay lines corresponds to the number of elements in the vector. ***Dline.m*** in turn calls the subrountine m-file ***cvdd.m*** which implements a "continuously variable digital delay element" [13]. This m-file filters the ***x*** input using an eight-tap Finite Impulse Response (FIR) filter whose tap coefficients are a function of the desired delay.

Later, the m-file ***ray_dop.m***, calculates the maximum Doppler shift frequency as a fraction of OFDM tone spacing as provided by the input, ***freqspace***. This m-file generates a random sequence of length L*N independent points of complex numbers with zero mean, and 0.5 variance real and imaginary parts. The envelope is Rayleigh with a mean square value of one. N is the number of FFT points. The amount of Doppler shifting is randomly calculated up to the maximum allowed using the seed parameter, ***s***, to set the seed of the random number generator. The real and imaginary parts are independently generated, and it is acceptable to enter a vector of Doppler shift values

equal to the number of delay paths. Additionally, the direct path is offset by 0.7 of the maximum input Doppler shift which is calculated by m-file ***ofst.m***. As a final step in ***chuhf.m***, the power losses for the individual multipaths are accounted for by multiplying each loss amount times the respective delay line output vectors. The output, ***y***, is a time domain representation of the transmitted signal plus multipath effects, presented as an array of complex received time domain samples.

3. Model 3 System

The COFDM model 3 system is depicted in Figure 32. In agreement with COFDM system model 1 and 2, the OFDM transmitter and OFDM receiver are identical. The only differences are in the channel of model 3.

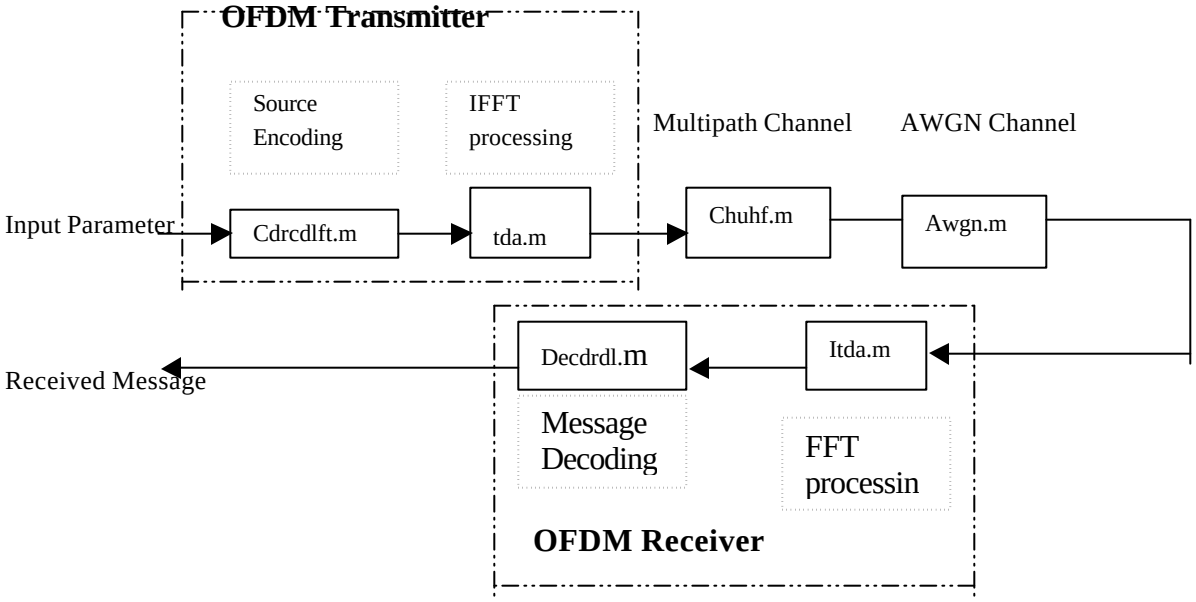


Figure 32. COFDM Model 3 System.

The channel 3 model consists of the channel 1 model (AWGN) combined with the channel 2 model (multipath) to form an overall complete channel 3 model. Both the channel 1 model and channel 2 model have been previously described in detail, implemented by m-files ***awgn.m*** and ***chuhf.m*** respectively. The channel 3 model is used extensively in system performance analysis presented in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SYSTEM SIMULATION METHODOLOGY AND TEST RESULTS

A. GENERAL TEST PLAN

After construction of the various system models and functional verification of the partially integrated sub-blocks are accomplished, the research progressed to the simulation test phase where complete integrated system simulation trials were performed using different channel models and the corresponding performance curves were generated. The general system simulation test plan is presented in Table 4 along with the associated m-files governing each respective test phase.

Test phase	Simulation	M-file
1	System Model 0	***Chn0cdl.m***
2.	System Model 1	***Cofdmsim.m***
3.	System Model 2	***Cofdmsim.m***
4.	System Model 3	***Cofdmsim.m***

Table 4. General Test Plan.

As indicated in Table 4, there are four independent test phases, advancing in the level of channel complexity starting from the easiest, channel model 0, to the most challenging and complex, channel model 3. Throughout the collection and evaluation of simulation data, the hierarchical test approach from simple to complex allows for careful study and evaluation of each channel model output individually.

B. TEST PHASE 1 – SYSTEM MODEL 0 SIMULATIONS

Initial system model 0 simulations are performed to verify proper integration of all system sub-blocks and to ensure a correctly working overall model. Recall that the COFDM model 0 system incorporates the channel 0 model, representing a perfect noise free channel without AWGN and multipath distortions. Hence, this model can be viewed simply as the OFDM transmitter output connected directly to the OFDM receiver input with no intervening channel block. With the prior assumption that the transmitter and receiver are functioning correctly according to design, then the source and sink message blocks should have identical content without symbol errors since there can not be any channel noise influences corrupting the signal. Consequently, any symbol error

occurrences in the sink message must be the result of an incorrectly implemented m-file program model.

With this mind, numerous system model 0 simulation were repeatedly conducted using m-file ***chn0cdl.m*** with various input configurations, and the resulting data collected and evaluated. A table of sample results reflecting model 0 system simulations with various input configurations is presented in Table 5.

```
» chn0cdl(0,0,0,0,222,4,4,6,8,4,4,8,8,8,6,0) % Block interleaver selected, random seed of 222 is chosen, 4 OFDM column and 6 rows. Guard interval of 6 is used. The number of FFT point used is 8. The M-ary number is 16.

Random_Source_Msg =

    5    4   13   12
    2    2   10    0
    3    4    1    9
    8    8    9   15
    2    6   15   13
   10    5    9   14
Sink_msg =

    5    4   13   12
    2    2   10    0
    3    4    1    9
    8    8    9   15
    2    6   15   13
   10    5    9   14
GREAT!!!there are no errors.
_____________________________________________
Test Passed!!!
```

Table 5. Model 0 Verification Example.

With the conclusion of transmitter and receiver functional verification, the remaining system test simulations including channel noise and multipath and are oriented around the channel 1, channel 2 and channel 3 models. Channel 3 simulates an actual indoor transmission environment, hence, it is the most indicative of the type of channel influences that will affect real-time RF communications during transmission by the WLAN.

C. TEST PHASE 2 – SYSTEM MODEL 1 SIMULATIONS

Test phase 2 performs channel 1 model simulations exclusively (AWGN channel) and compared the test results to [14]. In [14], the COFDM evalution was done by means

of computer simulations and it was implemented with DQPSK modulation, an AWGN channel, a convolutional rate of ½, and a constraint length ranging from 3 to 7. However, Viterbi soft decision decoding was used in [14] instead of the hard decision decoding that is adopted in this thesis. The BER performance presented in [14] is shown in Figure 33.

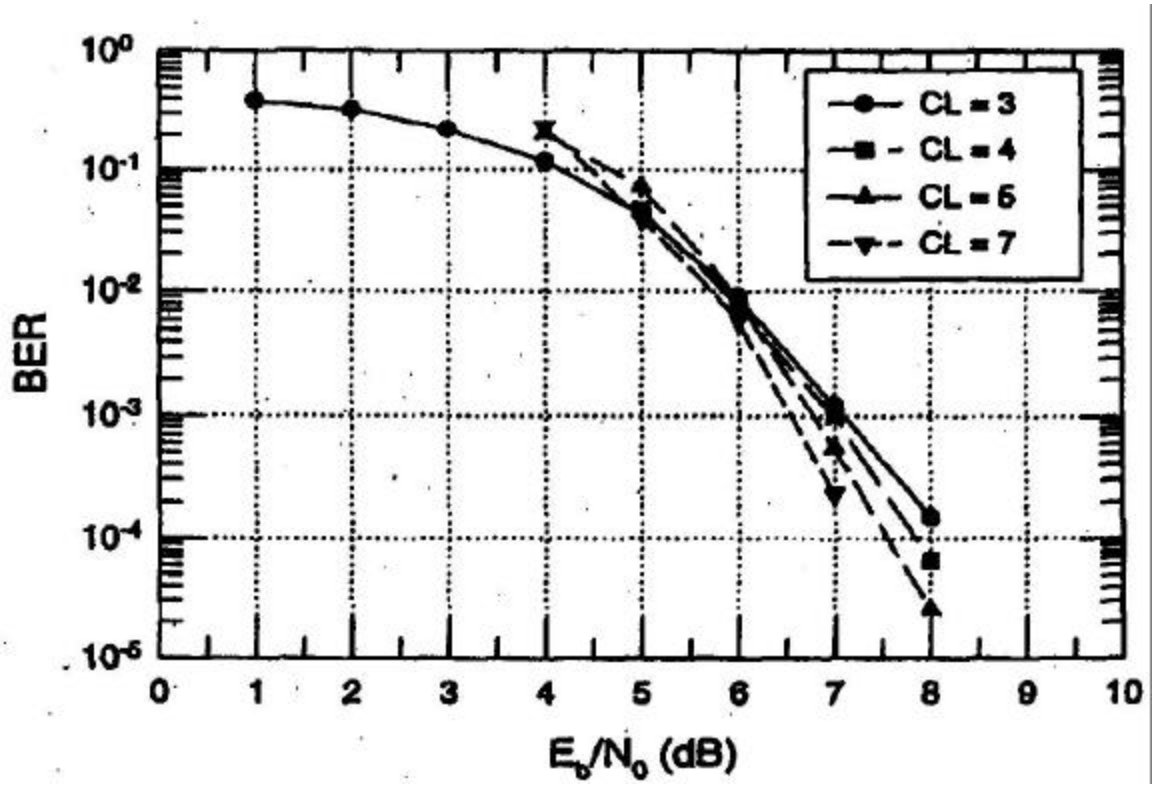


Figure 33. BER vs Eb/No For Different Constraint Length (CL) In AWGN Channel, After Thibault And Le, [14].

Recall that AWGN is emulated in MATLAB using the m-file ***awgn.m*** and is part of the COFDM model 1 system. During this test phase, the batch m-file ***cofdmsim.m*** is configured for system model 1 simulations and used to generate numerous test data sets. The data results are presented graphically in the form of performance curves representing the Bit Error Rates (BER) versus the ratio of bit energy to noise power density (E_b/N_o). Simulation data are compared to [14] COFDM AWGN performance curves (Figure 33) with similar system configurations. Evaluations of the results are conducted to measure the integrity of the system in the presence of AWGN.

As mentioned previously, the magnitude of each randomly generated message symbol and the corresponding complex modulation value are fixed at unity and represent the signal energies. However, the noise power density, N_o, is variable and configurable by the user. Consequently, during simulation configurations, selection of noise powers by setting suitable noise variance range (sigma parameter) promotes the generation of meaningful performance plots and allows for comparisons among various test configurations. Table 6 presents a portion of a ***cofdmsim.m*** simulation configured for system model 1 (AWGN channel) using 48 OFDM frequency tones and frequency differential encoding, per 802.11a, while Figure 34 depicts the corresponding performance plot associated with the configured inputs. Figure 35 and 36 show the transmitted signal and the effects of AWGN on the received signal.

```
» cofdmsim
_____________________________________________________________
This batch m-file runs COFDM simulations using different channel models.

To run the frequency version, enter 1(one), To run the time version, enter 0 (zero), or to run both enter
2(two):1

Enter the # of OFDM frequencies (note : must be even):48

Enter the number of FFT points (Note : This number must be larger than # of OFDM frequencies):64

Do you want to run channel model 0, channel model 1, channel model 2 or channel model 3 ? (Enter 0,1,2
or 3):1

Channel model 1 simulation performed.

Enter the sigma noise parameter range or single value. (Ex linspace(0,0.02,20)or
.003):[linspace(0,0.0542,30)]

Simulate all interleaver cases (yes) or specific ones(no)? (1=yes,0=no):0

Enter specific case numbers from (0 to 8)(Ex [0 4 5 8]):0

Enter the total minimum number of symbols to simulate (Ex 10000):20000

Note:Based on the parameters thus far, the actual total number of symbol to be simulated will be :20016
For the interleaver, do you want to calculate all possible intermediate matrix dimension
pairs?(1=yes,0=no):0

Desired interleaver pair? (Ex [row # col #] = [20 50] (Note: entering [1 20016],or [20016 1], offers no
interleaving functionality):[139 144]

Enter the number of M-ary bits, q (i.e. for 256-ary, q=8):1

Enter the number of N-ary bits,q(i.e. for 16-ary, q=4):2
```

Enter the guard interval length (Number of sample points):16

Enter specific seed values, or 0 for a random seed (ex [103 22, 60] or [0]):222

Do you want signal plots? (1=yes, 0=no):0

Do you want print outs? (1=yes, 0=no):0

Table 6. Model 1 Simulation Run Using Cofdmsim.m.

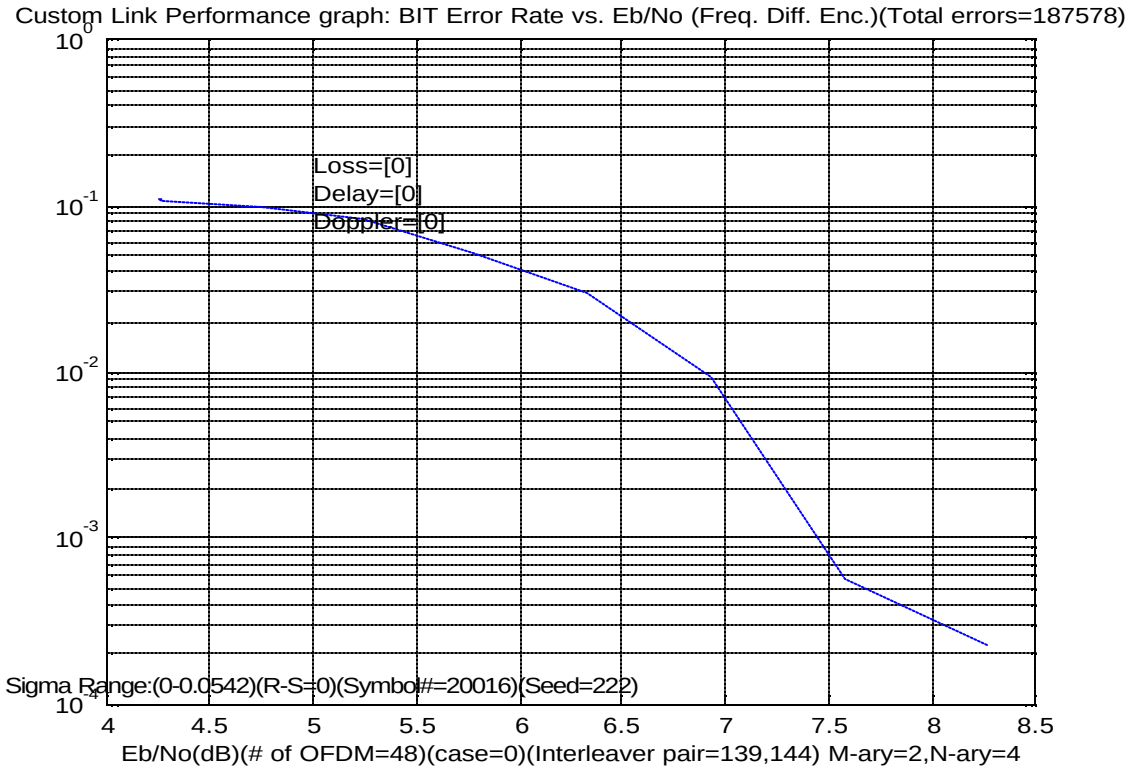


Figure 34. System Model 1 With AWGN Channel.

A corresponding received signal magnitude plot is depicted in Figure 36 with noticeable variation in the Received Signal Level. In contrast to the pre-transmitted magnitude plot (Figure 35), the noticeable signal variations in the received magnitude plot demonstrate the consequence of white Gaussian noise influences on the transmitted signal.

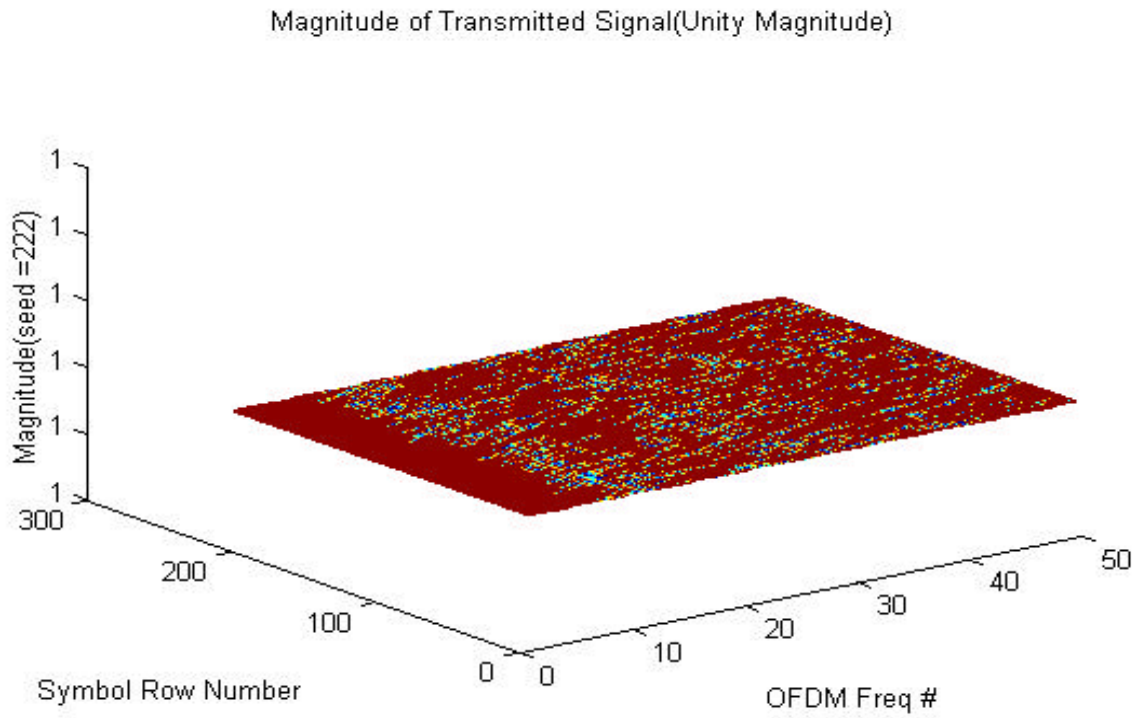


Figure 35. A Flat Planar Magnitude Representation Of Symbols Prior To Transmission.

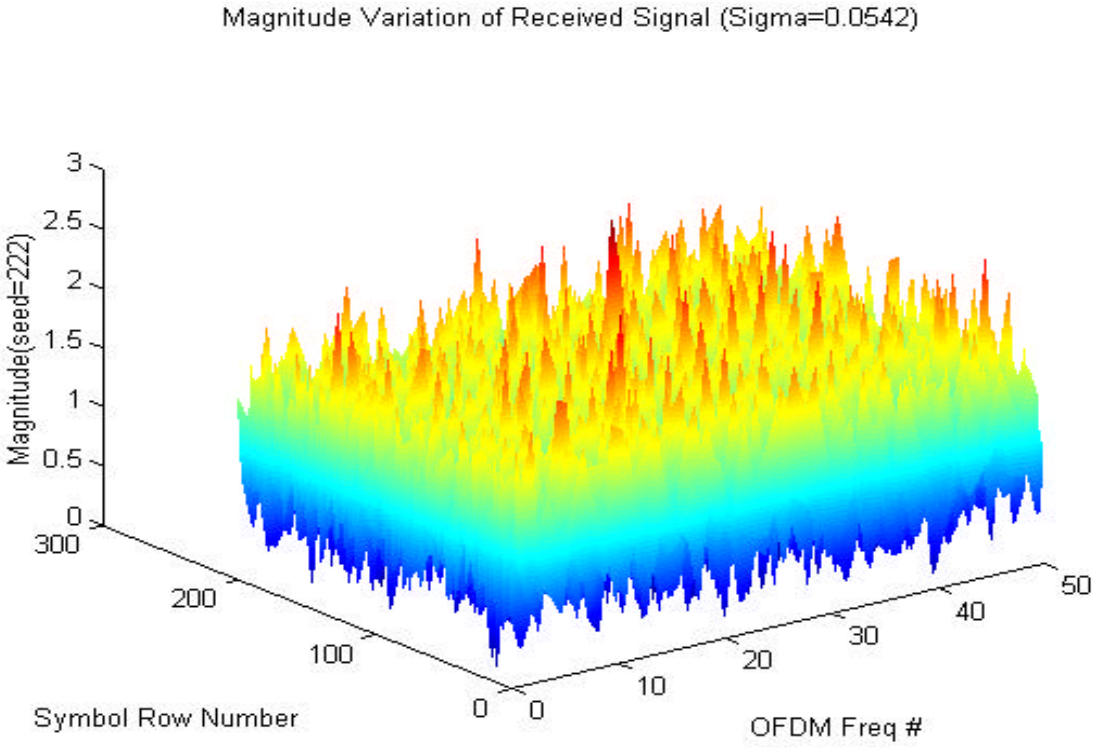


Figure 36. Effect Of AWGN Found On The Received Signal.

Comparison of the simulation BER performance graph in Figure 34 to the corresponding COFDM DQPSK graph shown in Figure 33 is depicted in Table 7 below.

P_b	Eb/No for simulation (Figure 34)	Eb/No for reference (Figure 33)	Difference, δ
10^{-2}	6.80	5.75	1.05
10^{-3}	7.40	6.50	0.90

Table 7. BERVs E_b/N_o : Comparison Of Simulated and Reference Plots (Figure 35).

The comparison in Table 7 shows that the simulated result is approximately 0.9dB (at $Pb=10^{-3}$) and 1.05dB (at $Pb=10^{-2}$) worse than [14]. The difference in performance may be due to Viterbi soft decision decoding that was used in [14] as compared to the hard decision decoding adopted in this thesis. A Viterbi decoder with soft decision data inputs quantized to three or four bits of precision perform better than one working with hard decision inputs [5].

It is apparent from phase 1 test result that system model 1 DQPSK simulation yields results similar to the BER performance in [14].

D. TEST PHASE 3 – SYSTEM MODEL 2 SIMULATIONS

The objective of this test phase is to simulate the system transmitting symbols through the multipath channel exclusively to reveal the burst error patterns. Phase 3 simulations were conducted using the batch file ***cofdmsim.m*** configured for system model 2 testing as shown in Table 8 below.

```
» cofdmsim

_____________________________________________________________

This batch m-file runs COFDM simulations using different channel models.

To run the frequency version, enter 1(one), To run the time version, enter 0 (zero), or to run both enter 2(two):1

Enter the # of OFDM frequencies (note : must be even):48

Enter the number of FFT points (Note : This number must be larger than # of OFDM frequencies):64

Do you want to run channel model 0, channel model 1, channel model 2 or channel model 3 ? (Enter 0,1,2 or 3):2
```

Channel model 2 simulation performed.
Do you want to run link1 ,link2, link3 or a custom link ? (Enter 1,2,3 or 4 for custom):1
Simulate all interleaver cases (yes) or specific ones(no)? (1=yes,0=no):0
Enter specific case numbers from (0 to 8)(Ex [0 4 5 8]):0
Enter the total minimum number of symbols to simulate (Ex 10000):10000
Note:Based on the parameters thus far, the actual total number of symbol to be simulated will be :10032
For the interleaver, do you want to calculate all possible intermediate matrix dimension pairs?(1=yes,0=no):0
Desired interleaver pair? (Ex [row # col #] = [20 50] (Note: entering [1 10032],or [10032 1], offers no interleaving functionality):[114 88]
Enter the number of M-ary bits, q (i.e. for 256-ary, q=8):1
Enter the number of N-ary bits,q(i.e. for 16-ary, q=4):2
Enter the guard interval length (Number of sample points):16
Do you want to include error correction coding ? (1=yes, 0=no):0
Enter specific seed values, or 0 for a random seed (ex [103 22, 60] or [0]):222
Do you want signal plots? (1=yes, 0=no):1
How many seconds of delay between pictures?1
Do you want print outs? (1=yes, 0=no):0

Table 8. Model 2 Simulation – Only Multipath Channel.

While performing ***cofdmsim.m*** system model 2 simulations, output plots depicting various forms of the signal data at strategic stages in the signal path are possible if desired and configured by the user. As an example of the types of plots generated during the simulation, Figure 37 through Figure 41 depict corresponding information generated by batch m-file ***cofdmsim.m*** configured as shown in Table 8.

Figure 37 depicts the constellation plot characteristic of DQPSK type modulation. As expected, 4 individual phase points are generated resulting from symbol mapping of 2-bit words into complex modulation values with unit magnitude and one of 4 possible phases. The constellation points, denoted by an asterisk, are symmetrically spaced on the unit circle, partitioning the circle into 4 equally sized sector formations.

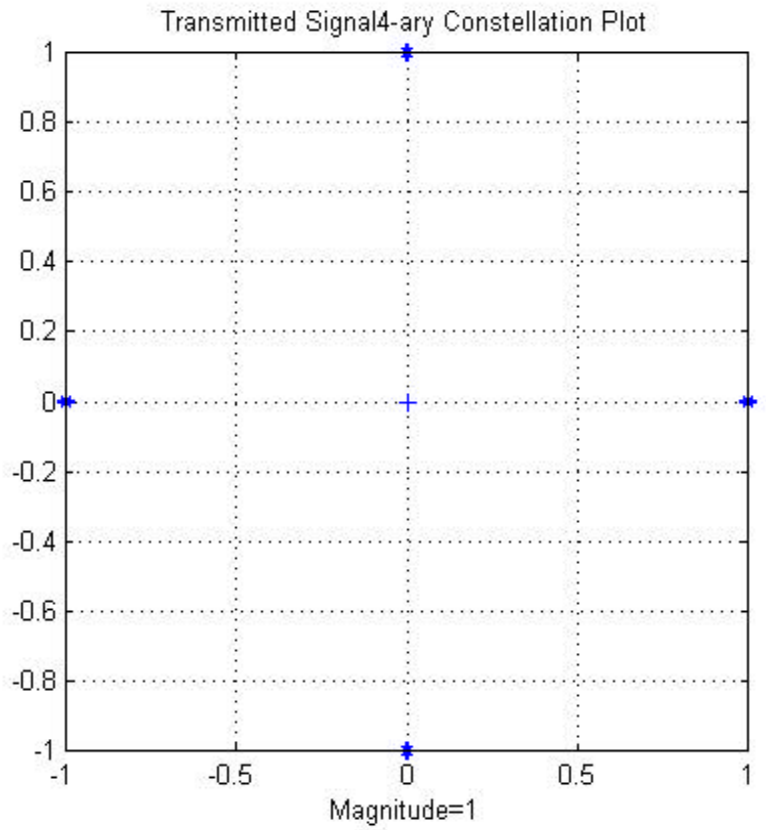


Figure 37. Constellation Plot Of DQPSK Modulation.

The corresponding message array of reformatted 2-bit PSK symbols with unit magnitude are depicted in Figure 38 and are transmitted through the channel. Notice the flat planar magnitude representation of the symbols prior to transmission. Recall that once a simulation is configured for a specified number of OFDM frequency tones, the number of tones remain fixed throughout the simulation duration. Consequently, additional symbols may be generated as a result of symbol word reformatting, increasing the original message array size in the time dimension (added symbol rows).

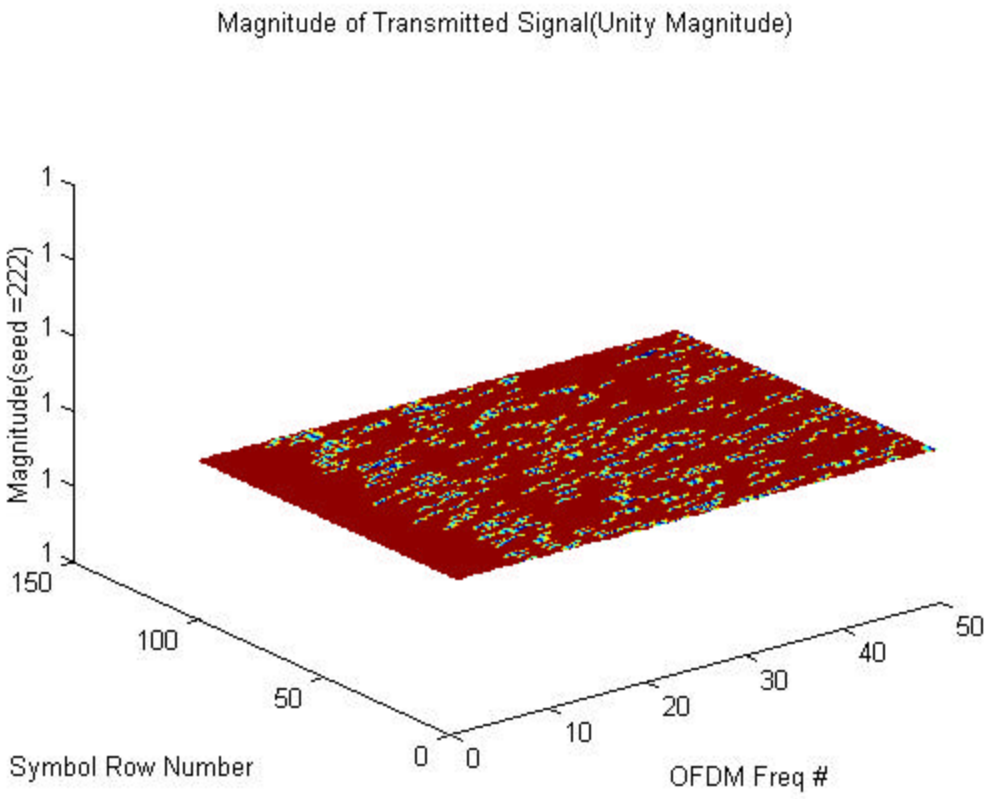


Figure 38. Transmitted Signal.

The corresponding received signal constellation plot is shown in Figure 39 and 40. As a consequence of multipath distortions within the channel causing constructive and destructive signal interference, the received constellation points are scattered from their normal pre-transmitted position (Figure 39). The figure also suggests that without additional signal conditioning, a majority of the received symbols would be decoded in error since many points cross sector borders into adjacent phase sectors. However, with the inclusion of differential encoding as demonstrated in Figure 40, the constellation points realign within their respective sector spaces forming a distinct star like structure.

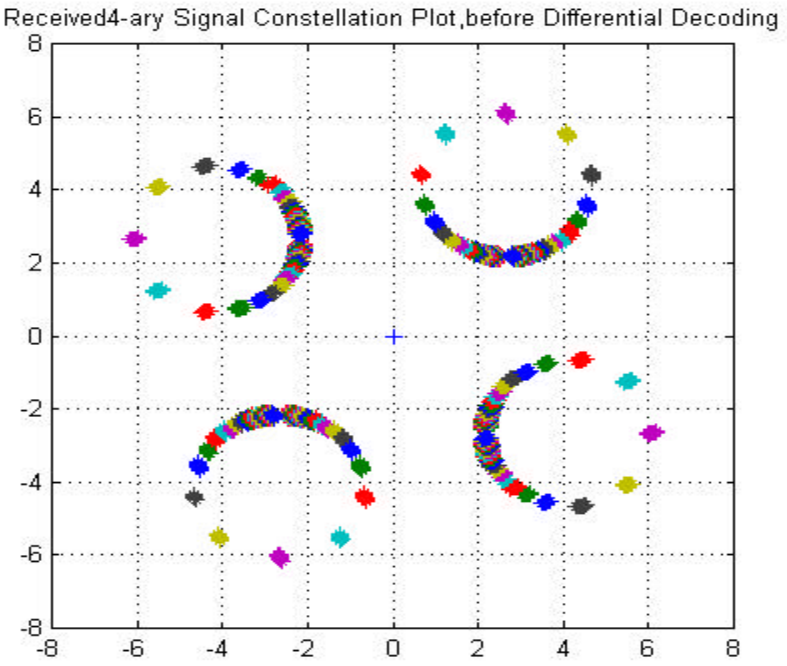


Figure 39. Signal Constellation Plot Before Differential Decoding.

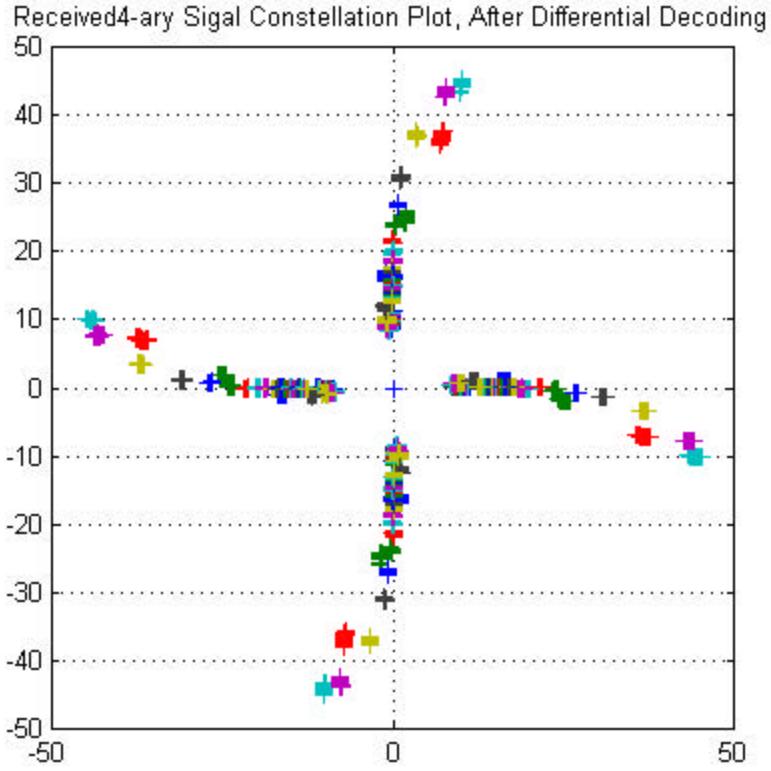


Figure 40. Signal Constellation Plot After Differential Decoding.

The corresponding received signal magnitude plot is depicted in Figure 41 with noticeable variations in the Received Signal Level (RSL), indicative of frequency selective fading.

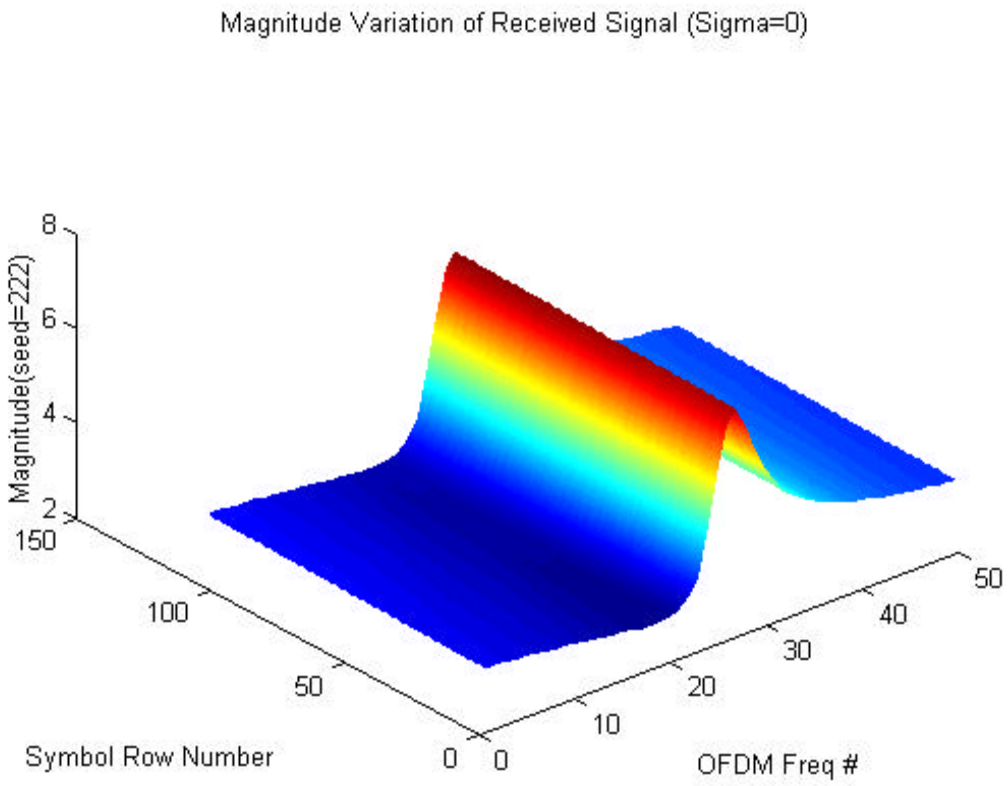


Figure 41. Received Signal With Frequency Selective Fading.

In stark contrast to the pre-transmitted magnitude plot (Figure 38), the noticeable peak and valley in the received magnitude plot demonstrate the consequences of multipath distortion influences on the transmitted signal through constructive and destructive signal interference by altering the message symbol magnitudes from their pre-transmitted unity levels. It is apparent that for this model 2 simulation, frequency selective fading occurs causing the frequency dependent peak and valley of the RSL.

E. TEST PHASE 4 – SYSTEM MODEL 3 SIMULATIONS

In this comprehensive test, complete system model 3 simulations are performed using channel 3 model (AWGN and multipath) to generate corresponding system performance curves. The multipath propagation model used in this test has been explained in Chapter 3 (Figure 20). There are total of 3 links created with different Doppler frequencies of 5, 10 and 15 Hz. Simulations were performed separately with these 3 links using the batch m-file ***cofdmsim.m***. A sample system model 3 simulation configuration using batch m-file ***cofdmsim.m*** is presented in Table 9.

```
» cofdmsim

_____________________________________________________________

This batch m-file runs COFDM simulations using different channel models.
To run the frequency version, enter 1(one), To run the time version, enter 0 (zero), or to run both enter 2(two):1
Enter the # of OFDM frequencies (note : must be even):48
Enter the number of FFT points (Note : This number must be larger than # of OFDM frequencies):64
Do you want to run channel model 0, channel model 1, channel model 2 or channel model 3 ? (Enter 0,1,2 or 3):3
Channel model 3 simulation performed
Enter the sigma noise parameter range or single value. (Ex linspace (0,0.02,20) or .003):[linspace(0,0.06,20)]
Do you want to run link1, link2, link3 or a custom link ?(Enter 1,2,3 or 4 for custom):1
Simulate all interleaver cases (yes) or specific ones(no)? (1=yes,0=no):0
Enter specific case numbers from (0 to 8)(Ex [0 4 5 8]):0
Enter the total minimum number of symbols to simulate (Ex 10000):20000
Note:Based on the parameters thus far, the actual total number of symbol to be simulated will be :20016
For the interleaver, do you want to calculate all possible intermediate matrix dimension pairs?(1=yes,0=no):0
Desired interleaver pair? (Ex [row # col #] = [20 50] (Note: entering [1 20016],or [20016 1], offers no interleaving functionality):[1668 12]
Enter the number of M-ary bits, q (i.e. for 256-ary, q=8):1
Enter the number of N-ary bits,q(i.e. for 16-ary, q=4):2
Enter the guard interval length (Number of sample points):16
Enter specific seed values, or 0 for a random seed (ex [103 22, 60] or [0]):33
```

Do you want signal plots? (1=yes, 0=no):1 How many seconds of delay between pictures?1 Do you want print outs? (1=yes, 0=no):0

Table 9. Model 3 Link 1 Simulation.

1. Link 1 With Doppler Frequency of 5 Hz

For this example, link 1 with a Doppler frequency of 5 Hz is used along with noise variance range of 0 to 0.06. The batch file generates performance curves similar to the ones presented during test phase 1, however, the performance is greatly degraded from AWGN theoretical curves due to the added multipath influences. The effect of the AWGN and multipath on the received signal is shown in Figure 42. Furthermore, an additional overhead loss of 25% from the inclusion of a 16 sample point guard interval precursor with 64 FFT points (16/64 = 0.25) reduces the effective information rate (Figure 43).

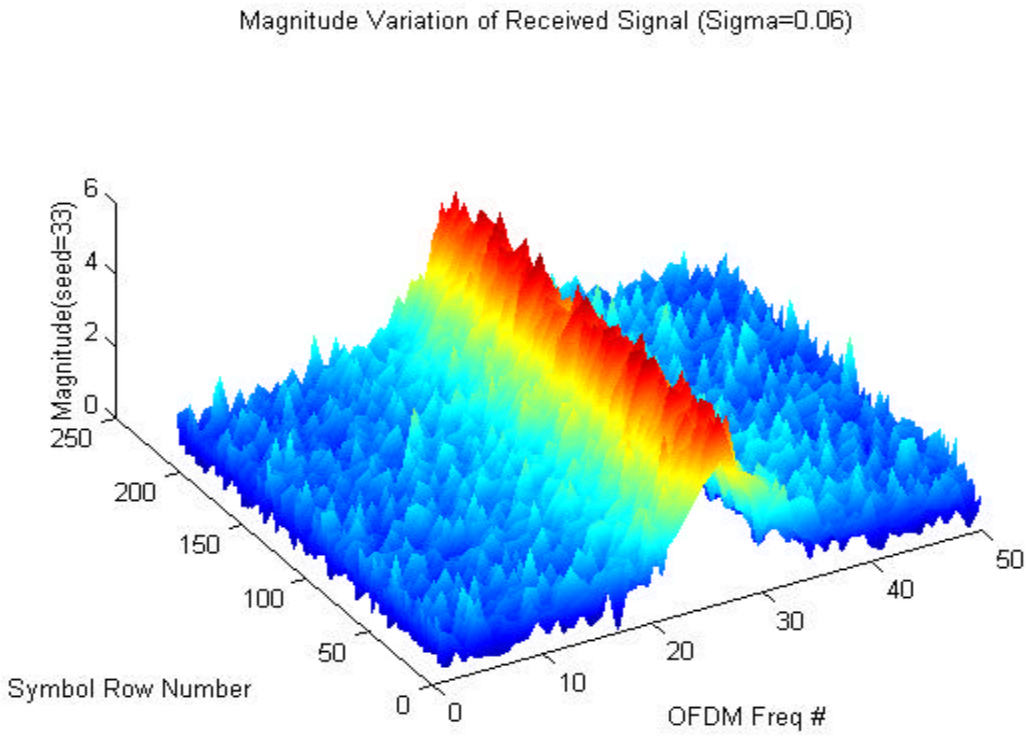


Figure 42. Effect of AWGN and Multipath On the Received Signal.

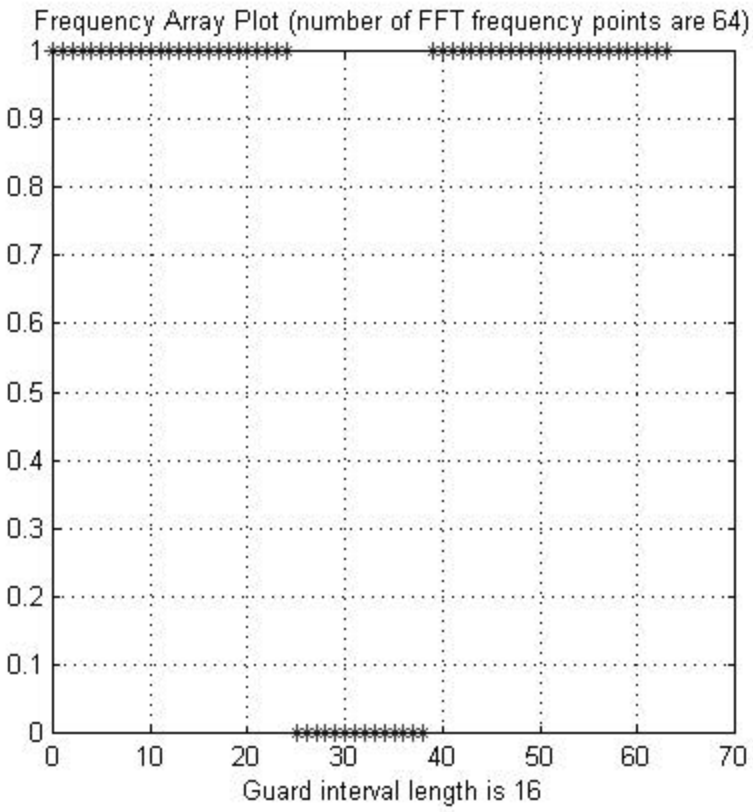


Figure 43. 16 Sample Point Guard Interval Precursor With 64 FFT Points.

The corresponding performance curve of the configured simulation in Table 9 is presented in Figure 44. From the plots in Figure 44 and Figure 33, the difference in performance is tabulated in Table 10 below.

P_b	Eb/No for Simulation Plot (Figure 44)	Eb/No for Reference Plot (Figure 33)	Difference, δ	Difference after correction for soft vs hard decision decoding
10^{-2}	8.65	5.75	2.90	2.90 - 1.05 = 1.85
10^{-3}	10.20	6.50	3.70	3.70 - 0.90 = 2.80

Table 10. BER vs. E_b/N_o : Comparison Of Simulated (Figure 44) And Reference Plots.

The comparison in Table 10 shows that model 3 link 1 which is subjected to the influence of AWGN and multipath, requires 2.9 to 3.7dB more than [14]. Recall that the result in [14] is exclusively due to AWGN only. Compensating for the difference in soft and hard Viterbi decoding decision used, we can deduce that the dB required for

multipath with link 1 (Doppler Frequency of 5 Hz) is between 1.85dB (at 10^{-2}) and 2.80dB (at 10^{-3}).

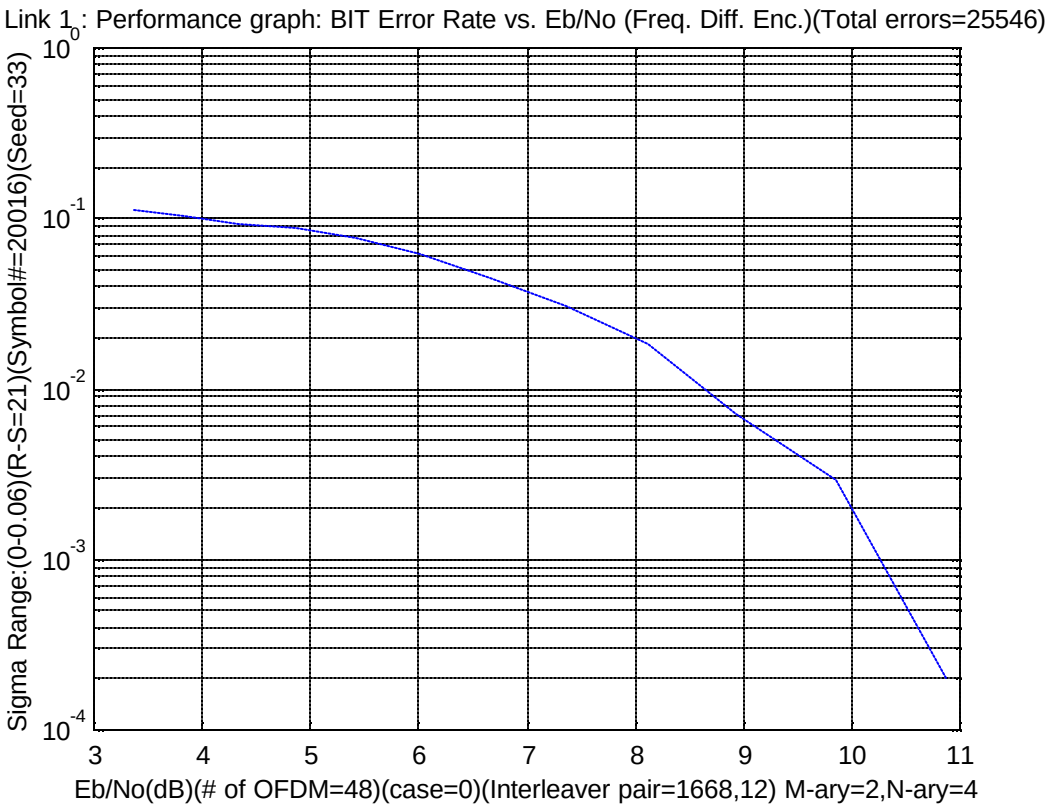


Figure 44. BER vs. E_b/N_o Performance - With Multipath & AGWN For Link 1.

2. Link 2 With Doppler Frequency of 10 Hz

The batch m-file ***cofdmsim.m*** was repeated, and link 2 was selected along with noise variance range of 0 to 0.06. The BER performance curve was obtained as shown in Figure 45. From the plot in Figure 45 and in comparison to Figure 33, the difference in performance is tabulated in Table 11.

P_b	Eb/No for Simulation Plot (Figure 45)	Eb/No for Reference Plot (Figure 33)	Difference, δ	Difference after correction for soft vs hard decision decoding
10^{-2}	8.60	5.75	2.85	2.85 - 1.05 = 1.80
10^{-3}	10.25	6.50	3.75	3.75 - 0.90 = 2.85

Table 11. BER vs. E_b/N_o : Comparison Of Simulated (Figure 45) And Reference Plots.

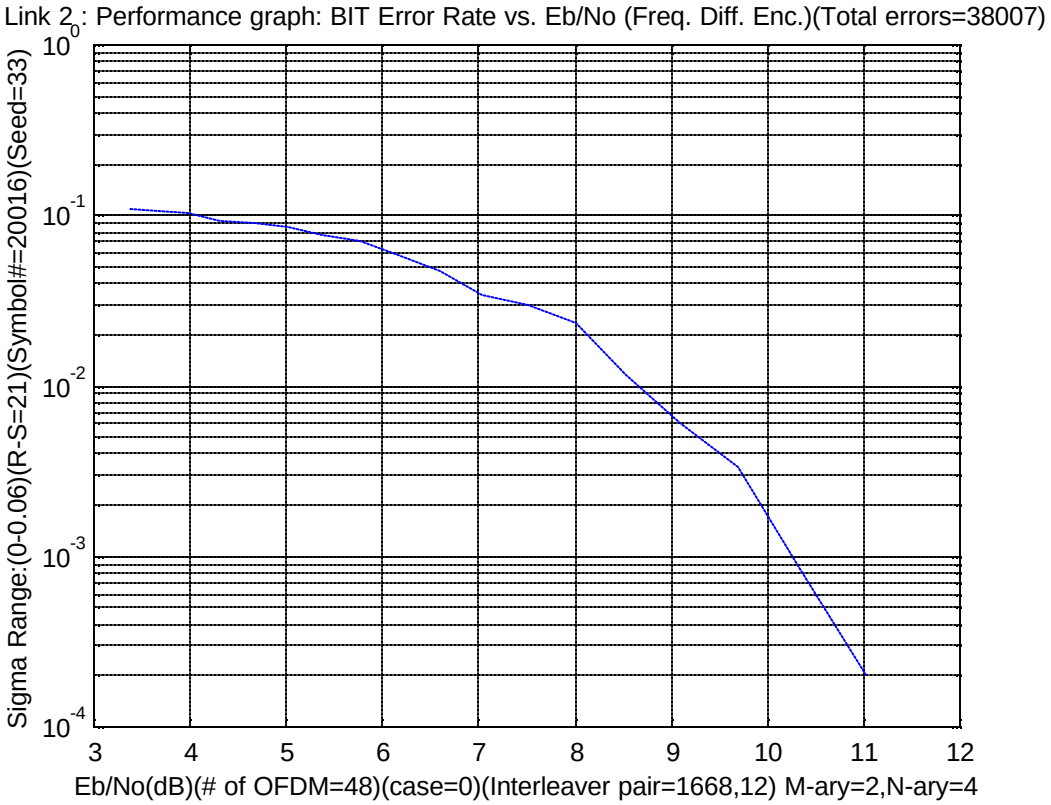


Figure 45. BER vs. E_b/N_o Performance - With Multipath & AGWN For Link 2.

The comparison in Table 11 shows that the difference in signal power due to multipath with link 2 (Doppler Frequency of 10 Hz) is between 1.80dB (at 10^{-2}) and 2.85dB (at 10^{-3}). The result obtained is very close to that achieved for link 1.

3. Link 3 With Doppler Frequency of 15 Hz

The BER performance curve for link 3 was obtained by running the batch m-file ***cofdmsim.m*** along with noise sigma parameter range of 0 to 0.06. The BER performance curve is shown in Figure 46. From the plots in Figure 46 and Figure 33, the difference in performance is tabulated in Table 12.

P_b	Eb/No for Simulation Plot (Figure 46)	Eb/No for Reference Plot (Figure 33)	Difference, δ	Difference after correction for soft vs hard decision decoding
10^{-2}	8.60	5.75	2.85	2.85 - 1.05 = 1.80
10^{-3}	10.15	6.50	3.65	3.65 - 0.90 = 2.75

Table 12. BER vs. E_b/N_o: Comparison Of Simulated (Figure 46) And Reference Plot.

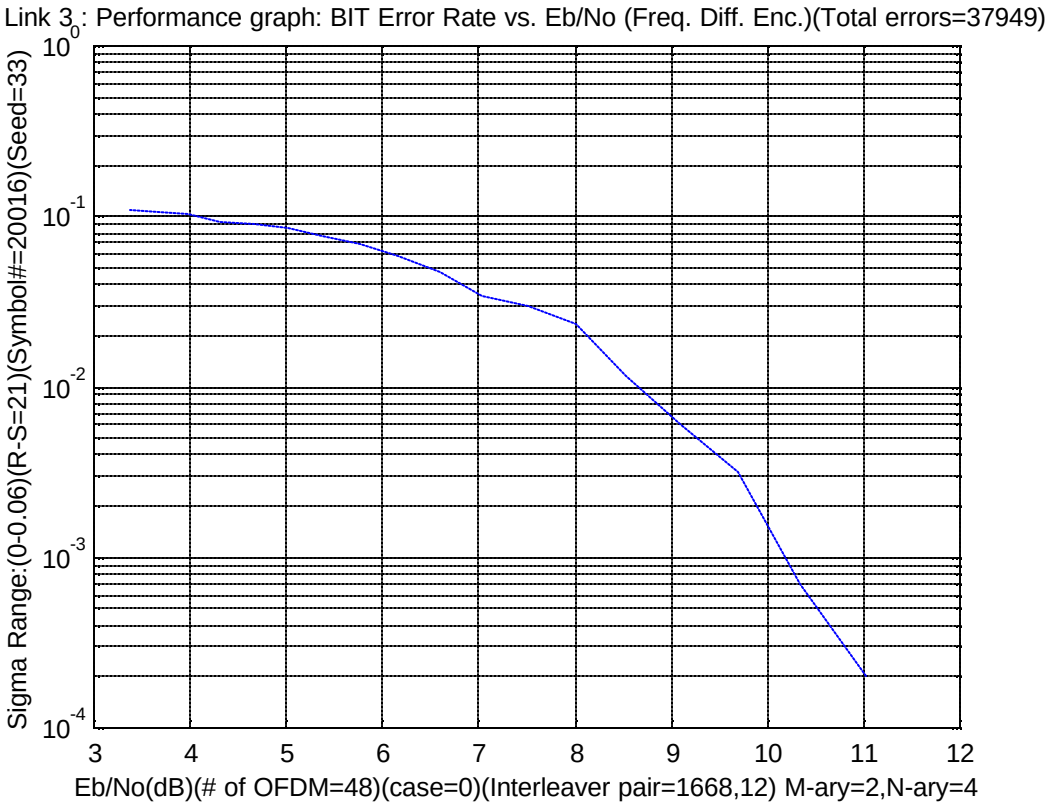


Figure 46. BER vs. E_b/N_o Performance - With Multipath & AGWN For Link 3.

The comparison in Table 12 shows that the difference in signal power due to multipath with link 3 (Doppler Frequency of 15 Hz) is between 1.80dB (at 10^{-2}) and 2.75dB (at 10^{-3}). Again the result obtained is similar with that achieved for link 1 and 2.

The results obtained for links 1 to 3 show that this COFDM configuration is immune to Doppler shift of 5 to 15 Hz. It is known that additional Doppler shifting causes symbol spectra and their respective sub-carriers to shift their frequency location into adjacent symbol areas causing spectral overlap. Our COFDM configuration uses only 48 tones, thus it offers good Doppler immunity since the frequency spacing is larger.

In [16], the maximum Doppler shift, f_{dm} in Hz is defined as :

$$f_{dm} = \frac{v}{l} = 1.4815\, f_G v_{mph} \qquad (6\text{-}1)$$

Where V_{mph} is the speed expressed in m.p.h, and the radio frequency, f_G is in GHz. Hence with 15Hz of Doppler shift, and assuming f_G used is 5.2GHz per 802.11a then the actual speed is :

$$15 = 1.4815*5.2*V_{mph} \quad (6\text{-}2)$$

$$V_{mph}=1.95 \text{ mph} = 0.87\text{m/s} \quad (6\text{-}3)$$

Doppler Frequency (Hz)	Speed (m/s)
5	0.29
10	0.58
15	0.87

Table 13. The Equivalent Speed For Doppler Frequencies Of 5, 10 And 15.

The above Doppler Shifts would be seen only if the velocity is entirely in the radial direction. The above Doppler frequencies used are all less than 1m/s which are a good representation of a human's walking speed in an indoor environment [11]. Hence we can further deduce that this COFDM configuration is robust enough to withstand the indoor mobility requirements.

F. PERFORMANCE OF COFDM WITH DBPSK MODULATION

After successful implementation of COFDM with DQPSK modulations, it is also desirable to examine the COFDM's performance using DBPSK modulation. The following simulations were conducted to evaluate the configuration performance under the influence of AWGN (exclusively) and combination of both AWGN and multipath effects :

- Test 1 - Model 1 with Only AWGN channel is used.
- Test 2- Model 3 using link 1 with 5Hz Doppler frequency.
- Test 3 - Model 3 using link 2 with 10Hz Doppler frequency.
- Test 4 – Model 3 using link 3 with 15Hz Doppler frequency.

The BER performance curves for the above tests are shown from Figure 47 to 50. The test results are tabulated in Table 14, and comparisons are made against DQPSK's performances.

Test	P_b	E_b/N_o for DBPSK Simulation	E_b/N_o for DQPSK Simulation	Difference, δ
Model 1 With AWGN	10^{-2}	5.75	6.80	-1.05
	10^{-3}	6.75	7.40	-0.65
Model 3 With Link 1	10^{-2}	7.30	8.65	-1.35
	10^{-3}	8.40	10.20	-1.80
Model 3 With Link 2	10^{-2}	7.30	8.60	-1.30
	10^{-3}	8.40	10.15	-1.75
Model 3 With Link 3	10^{-2}	7.00	8.60	-1.60
	10^{-3}	8.40	10.25	-1.85

Table 14. DBPSK vs. DQPSK.

The above comparisons show that DBPSK required less E_b/N_o then DQPSK. Under the influence of AWGN and multipath, the DBPSK modulations show that the E_b/N_o required for links 1 to 3 simulations are similar, except link 3 at 10^{-2} probability which is 0.30dB less.

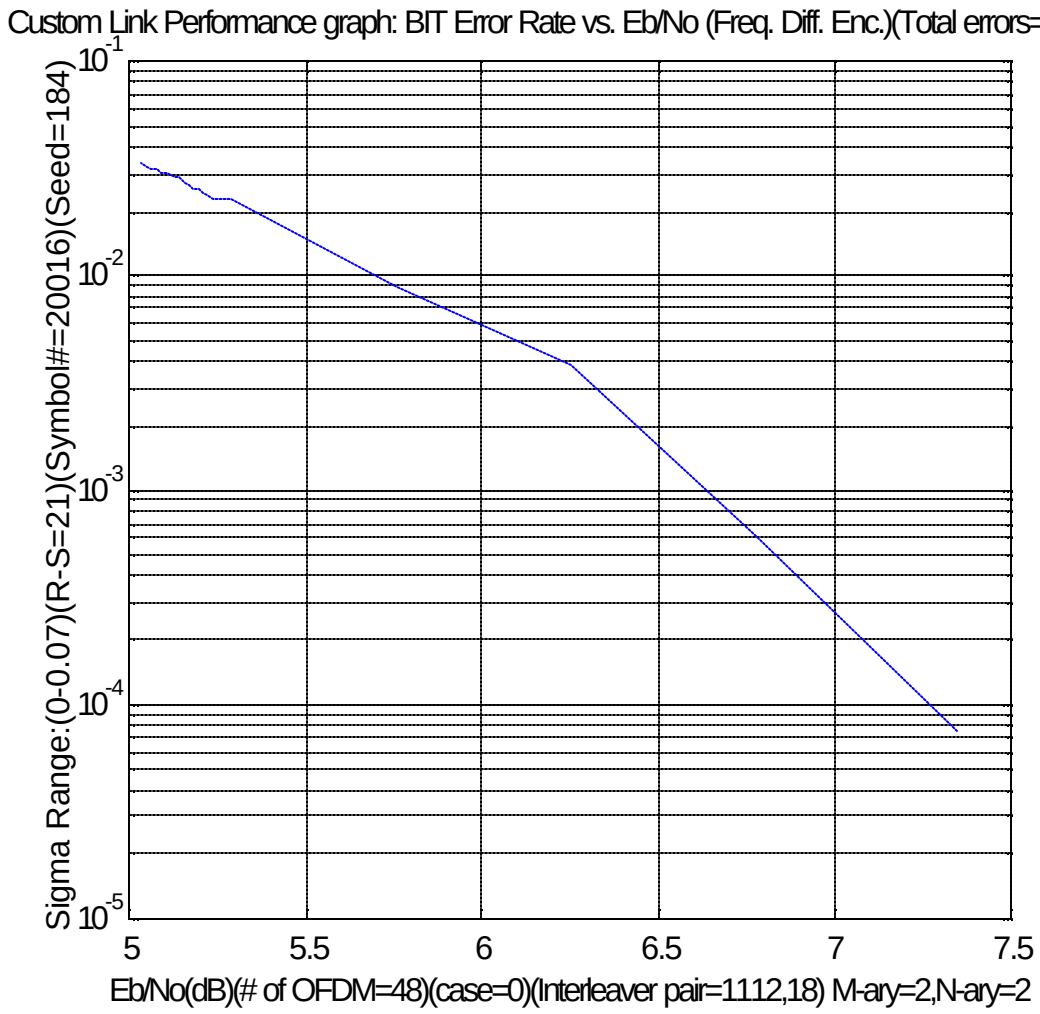


Figure 47. COFDM DBPSK Modulation With Model 1 (AWGN).

Figure 48. COFDM DBPSK Modulation With Model 3 Link 1.

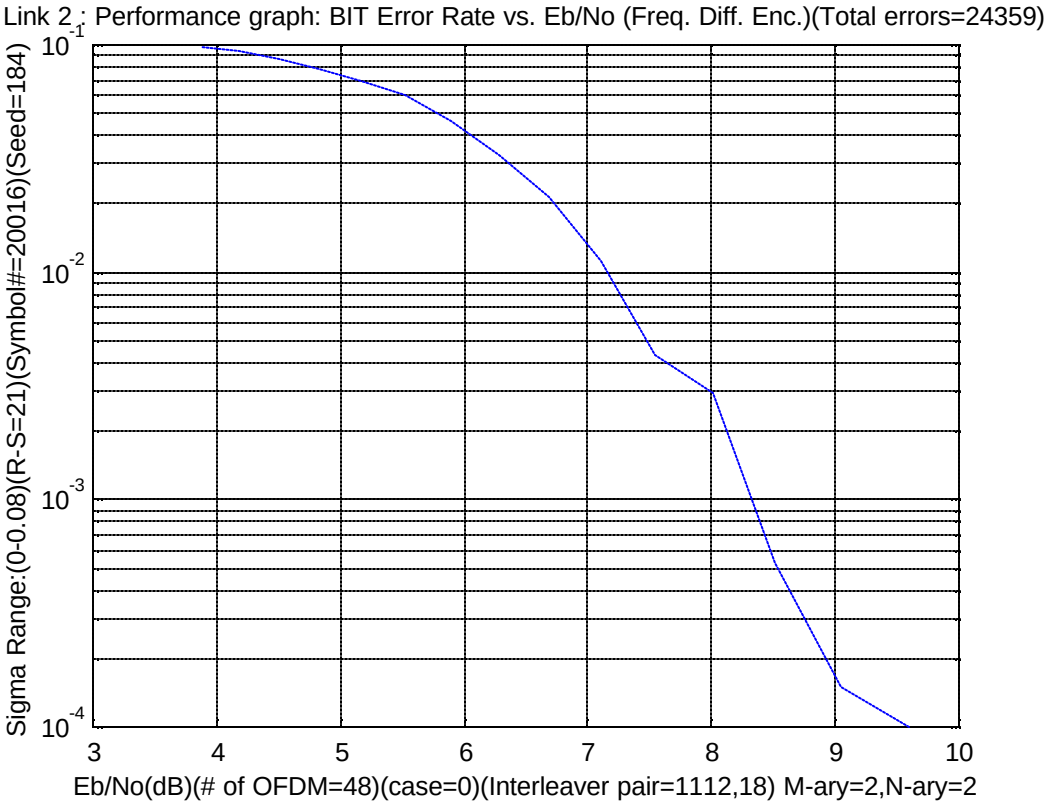


Figure 49. COFDM DBPSK Modulation With Model 3 Link 2.

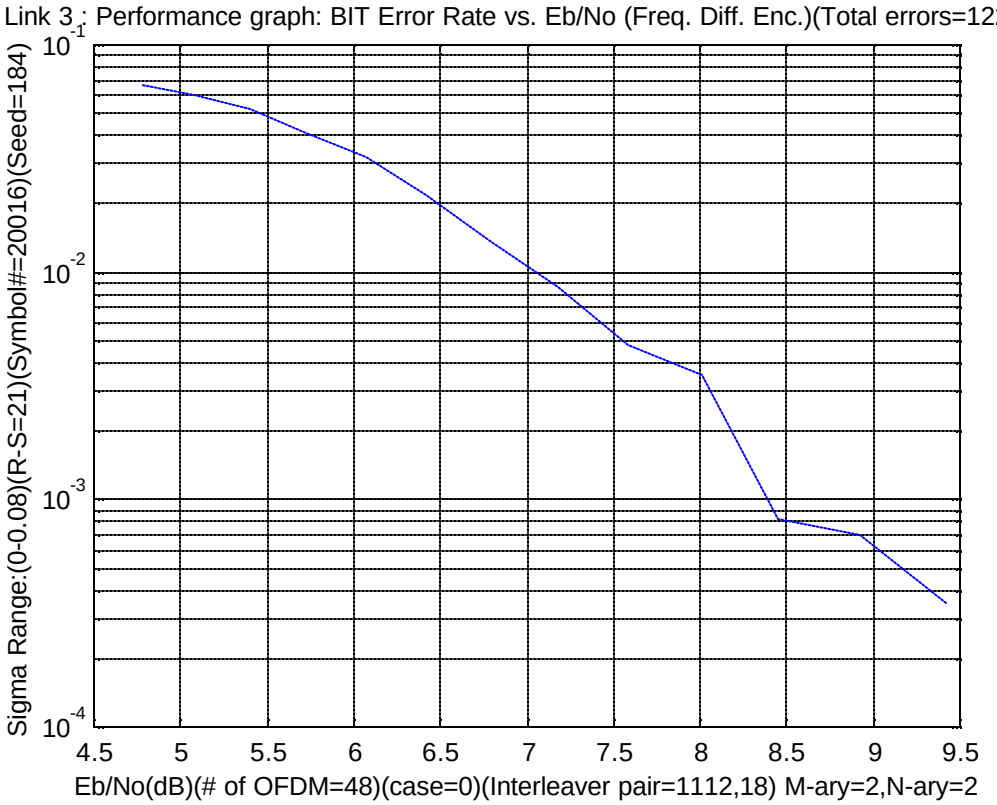


Figure 50. COFDM DBPSK Modulation With Model 3 Link 3.

VII. CONCLUSIONS

A. DISCUSSION OF SIMULATION RESULTS

The objective of simulating the physical layer of the IEEE 802.11a has been successfully achieved in this thesis. The simulation results showed that COFDM system is capable of indoor environment communications in the presence of known multipath and noise conditions. Further discussions relating to specific test phases are presented below.

1. Test Phase 1 and Test Phase 2 Discussions

Test phase 1 validated a functionally correct model, as there was an absence of errors in the sink message with no channel included. This indicated that all system sub-blocks within the transmitter and the receiver were operating correctly according to design, and no obvious design flaws existed due to inaccurate m-file construction. Test phase 2 carries the functional verification one step further by also including complete system model 1 simulations (with AWGN only). This test permits performance curve comparisons to [14] AWGN curves for DQPSK to further verify correct simulation. Results of system simulations indicated that system model 1 performance is approximately 0.9dB (at $Pb=10^{-3}$) and 1.05dB (at $Pb=10^{-2}$) worse than [14]. The difference in performance may be due to Viterbi soft decision decoding that was used in [14] as compared to the hard decision decoding adopted in this thesis. A Viterbi decoder with soft decision data inputs quantized to three or four bits of precision perform better than one working with hard decision inputs [5].

2. Test Phase 3 Discussions

Test phase 3 simulation using the channel 2 model (multipath channel only) exclusively demonstrated the effects of multipath on the received signal and the corresponding sink message array error event manifestations. As expected, frequency selective fading occurred as well as partial flat fading. This test phase was also useful in depicting the behaviors of the received signal magnitudes and phases as seen by the

constellation and magnitude plots. As anticipated, these plots demonstrated constructive and destructive interference due to channel multipath distortions, as evident by the distinguishing peak and valley apparent in the received signal magnitude plots. The received constellation plots demonstrated the manner in which individual symbol signal points were shifted in phase from their characteristic pre-transmitted positions.

3. Test Phase 4 Discussions

Test phase 4 involved comprehensive testing of a complete system model 3 simulations using channel 3 model (AWGN and multipath) to generate corresponding system performance curves. The multipath propagation model used in this test has been explained in Chapter 3 (Figure 20). There are total of 3 links created with different Doppler frequency of 5, 10 and 15 Hz. Simulations were performed separately with these 3 links using the batch m-file ***cofdmsim.m***. In comparison to test phase 2 (AWGN only), the results showed that more power is required to combat the multipath effect. The extra power needed is between 1.80 to 1.85dB at 10^{-2} probability and between 2.75 to 2.80dB at 10^{-3} probability. The results obtained for links 1 to 3 also showed that the COFDM configuration is immune to a Doppler shift of 5 to 15 Hz. Since our COFDM configuration uses only 48 tones, it offers good Doppler immunity as the frequency spacing is larger. The above Doppler frequencies are all from a transmitter velocity of less than 1m/s which is a good representation of a human's walking speed in an indoor environment [11]. Hence, we can further deduce that this COFDM configuration is robust enough to withstand the indoor mobility requirements.

4. COFDM DBPSK Modulation Discussions

The COFDM configuration was further examined with DBPSK modulation. As expected, the results showed that DBPSK required less E_b/N_o than DQPSK. Under the influence of AWGN and multipath, the DBPSK modulation shows that the E_b/N_o required for links 1 to 3 simulations are similar.

B. FUTURE WORK

The research presented in this thesis has successfully demonstrated the COFDM performance in the presence of known AWGN and multipath conditions. However, it is noted that a soft decision Viterbi decoder with three or four bits of precision would perform better than the present one that is working with hard decision inputs. The MATLAB function ***viterbi.m*** implemented in this thesis can also be used for soft decision decoding of convolution codes. The separate file ***metric.m*** defines the metric used in the decoding process. For hard decision decoding, this metric uses the Hamming distance; for soft decision decoding, it is the Euclidean distance.

The code rate used for the convolutional encoder can also be increased from ½ to ¾ by employing "puncturing". Puncturing is a procedure for omitting some of the encoded bits in the transmitter (thus reducing the number of transmitted bits and increasing the coding rate) and inserting a dummy "zero" metric into the Viterbi decoder on the receiver side in place of the omitted bits.

Further work can explore replacing the Differential M-PSK with the M-ary Quadrature Amplitude Modulation (QAM). It is envisaged that the modification would mainly involve m-files such as ***difcdrft.m*** and ***dfdcdrft.m***. In addition, pilot tones or equalization must be used for QAM in mobile systems [11].

Finally, the performance curves obtained in this thesis have the potential for high visibility and impact in several operational projects [1]. The increasing prevalence of WLAN, both within the Defense establishment and in the public domain, underscores the need for a simulation of this kind. The results obtained from this thesis can be included into the Radio pipeline of OPNET simulation package. The OPNET version 7 comes with an IEEE 802.11 model, and it can be modified to function as an IEEE 802.11a

WLAN. Hence, the performance of this newly proposed IEEE 802.11a WLAN protocol in either an office or submarine environment can be completely analyzed.

APPENDIX A. COFDM MATLAB SOURCE CODE

AWGN

```
%function [Y]=awgn(X,s,N,sigma)
%-------------------------------------------------------------------------------
%
%Title              : Additive White Gaussian Noise (Channel Model 1)
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%INPUTS:
%           X       : Input array of time domain complex modulation values
%           s       : Seed parameter for random number generator
%           N       : Number of OFDM frequencies (FFT size),includes zero pad
%           sigma   : Noise parameter for calculating Eb/No (function of the noise
variance)
%
%OUTPUTS:
%           Y       : Output signal plus noise,array of time domain complex numbers
%-------------------------------------------------------------------------------
function Y = awgn(X,s,N,sigma)
%
%Find dimensions of the input array
[rr,cc]=size(X);
%
%Seed configurations to set the random# generator seed
randn('seed',s+30);
%
%Generate a random real part
wreal=randn(rr,cc);
%
%Generate a random imaginary part
randn('seed',s+40);
wimg=i*randn(rr,cc);
%
%An array of random complex entries chosen from a normal distribution with
%mean 0.0 and variance 1.0.  Array dimensions is the same as X.
W=wreal+wimg;
%
%Random noise multiplied by the sigma factor and added to the signal.
Y=X+(sigma.*W);
%-------------------------------------------------------------------------------
```

BIN2DECI

```
%function [vy]=bin2deci(vx)
%-------------------------------------------------------------------------------
%
%Title              : Binary To Decimal Conversion
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%INPUTS:
%          vx       : Binary inputs
%
%OUTPUTS:
%          vy       : Decimal output
%-------------------------------------------------------------------------------
function v_y=bin2deci(v_x)
v_l=length(v_x);
v_y=(v_l-1:-1:0);
v_y=2.^v_y;
v_y=v_x*v_y';
```

BM

```
%function [m]=bm(q,v)
%-------------------------------------------------------------------------------
%
%Title                  : Binary To M-ary Converter
%Thesis Advisor         : Prof J. McEachen, Naval Postgraduate School
%Author                 : Prof. Paul H. Moose, Naval Postgraduate School
%Modified       by      : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%INPUTS:
%               q       : Base 2 exponent for M-ary symbol generation
%               v       : Binary data vector
%
%OUTPUTS:
%               m       : M-ary output vector in decimal notation
%-------------------------------------------------------------------------------
function m=bm(q,v)
%
%Find the length of input vector,v,and determine if there is a remainder
%after dividing by q
n=length(v);
r=rem(n,q);
%
%If there is no remainder,don't pad v input vector.  Otherwise add the appropriate
%number of zeros to generate a code word with an exact multiple of q bits.
%
if r==0
  v=v;
else
  v=[v zeros(1,q-r)];
end
%
%Place least significant bit of the symbol on the left end.

map=1;
for j=1:q-1
  map=[map 2^j];
end
%
%Remove q bits at a time from v to generate m-ary symbol values.

n=length(v);
p=round(n/q);
A=zeros(q,p);
```

```
A(:)=v;
m=map*A;
m_ary_msg=m;
%--------------------------------------------------------------------------------
```

CDLDLV

```
%function [s]=cdldlv(l,k,case,si,SYNC)
%-------------------------------------------------------------------------------
%
%Title              : CDL Block Deinterleaver
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%INPUTS:
%          l        : Number of rows in intermediate matrix
%          k        : Number of columns in intermediate matrix
%          case     : Variable indicating the deinterleaving method to be
%                     used (9 different cases)
%          si       : Input message string to be deinterleaved
%          sinc     : Frame synchronization bits (Not used in COFDM simulation)
%
%OUTPUTS:
%          s        : Interleaved output string
%-------------------------------------------------------------------------------
function s=cdldlv(l,k,case,si,SYNC)
si(length(si)+l-length(SYNC):length(si))=zeros(l,length(SYNC));
N=length(si);
if l*k==N
  x=zeros(l,k);
  x(:)=si;
  K=(1:k)-1;
  CR=K.*(K+1)/2;
  L=(1:l)-1;
  RR=L.*(L+1)/2;
  %
  if case==1
    for kk=1:k
      x(:,kk)=rotm(x(:,kk),CR(kk));
    end
  elseif case==2
    for kk=1:k
      [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
    end
  elseif case==3
    for kk=1:l
      x(kk,:)=rotm(x(kk,:),RR(kk));
    end
  elseif case==4
```

```
    for kk=1:l
      [z,x(kk,:)]=rotm(x(kk,:),RR(kk));
    end
  elseif case==5
    for kk=1:k
      x(:,kk)=rotm(x(:,kk),CR(kk));
    end
    for ll=1:l
      x(ll,:)=rotm(x(ll,:),RR(ll));
    end
  elseif case==6
    for kk=1:k
      [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
    end
    for ll=1:l
      x(ll,:)=rotm(x(ll,:),RR(ll));
    end
  elseif case==7
    for kk=1:k
      x(:,kk)=rotm(x(:,kk),CR(kk));
    end
    for ll=1:l
      [z,x(ll,:)]=rotm(x(ll,:),RR(ll));
    end
  elseif case==8
    for kk=1:k
      [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
    end
    for ll=1:l
      [z,x(ll,:)]=rotm(x(ll,:),RR(ll));
    end
  end
  x=x';
  s=x(:);
  s=s';
end
%-------------------------------------------------------------------------------
```

CDLILV

```
%function si=cdlilv(l,k,case,s,SYNC)
%-------------------------------------------------------------------------------
%
%Title              : CDL Block Interleaver
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Prof. Paul H. Moose, Naval Postgraduate School
%Modified      by   : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%INPUTS:
%           l       : Number of rows in intermediate matrix
%           k       : Number of columns in intermediate matrix
%           case    : Variable indicating the deinterleaving method to be
%                     used (9 different cases)
%           s       : Input message string to be deinterleaved
%           SYNC    : Frame synchronization bits (Not used in COFDM simulation)
%
%OUTPUTS:
%           si      : Interleaved output string
%
%Subroutines Used : rotm.m
%-------------------------------------------------------------------------------
function si = cdlilv(l,k,case,s,SYNC)
N=length(s);
if l*k==N
  x=zeros(l,k);
  x=x';
  x(:)=s;
  x=x';
  Intermediate_mx=x;
  K=(1:k)-1;
  CR=K.*(K+1)/2;
  L=(1:l)-1;
  RR=L.*(L+1)/2;
  %
  if case==1
    for kk=1:k
      [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
    end
  elseif case==2
    for kk=1:k
      x(:,kk)=rotm(x(:,kk),CR(kk));
    end
  elseif case==3
```

```
    for kk=1:l
      [z,x(kk,:)]=rotm(x(kk,:),RR(kk));
    end
  elseif case==4
    for kk=1:l
      x(kk,:)=rotm(x(kk,:),RR(kk));
    end
  elseif case==5
    for ll=1:l
      [z,x(ll,:)]=rotm(x(ll,:),RR(ll));
    end
    for kk=1:k
      [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
    end
  elseif case==6
    for ll=1:l
      [z,x(ll,:)]=rotm(x(ll,:),RR(ll));
    end
    for kk=1:k
      x(:,kk)=rotm(x(:,kk),CR(kk));
    end
  elseif case==7
    for ll=1:l
      x(ll,:)=rotm(x(ll,:),RR(ll));
    end
    for kk=1:k
      [z,x(:,kk)]=rotm(x(:,kk),CR(kk));
    end
  elseif case==8
    for ll=1:l
      x(ll,:)=rotm(x(ll,:),RR(ll));
    end
    for kk=1:k
      x(:,kk)=rotm(x(:,kk),CR(kk));
    end
  end
  si=x(:);
  si=si';
end
si(length(si)-length(SYNC)+1:length(si))=SYNC;
%-------------------------------------------------------------------------------------
```

CDRCDLFT

```
%function
[Fa,MD,B_ce,B_random,nsymno]=cdrcdlft(picy_n,pic,case,s,freqno,rintlv,cintlv,N,mary,
nary,fort)
%-------------------------------------------------------------------------------
%
%Title              : COFDM Encoder with CDL Interleaver
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Prof. Paul H. Moose, Naval Postgraduate School
%Modified      by   : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%INPUTS:
%          picy_n  : Switch variable to allow or disallow the generation of figures
%          pic     : Argument passed by another calling m-file to indicate the loop
number
%          s       : Seed parameter for random number generator
%          freqno  : Number of OFDM frequencies (sub-carriers) used in each message
array
%          rintlv  : Interleaver parameter for intermediate matrix row #
%          cintlv  : Interleaver parameter for intermediate matrix column #
%          N       : Number of FFT frequency sample points,must be larger than freqno
%          mary    : Initial M-ary symbol format (OFDM symbol bit length)
%          nary    : Final N-ary symbol format (PSK symbol bit length)
%          fort    : Selects either frequency (fort=1) or time (fort=0) differential
encoding
%
%OUTPUTS:
%          Fa      : Frequency array of prearranged modulation values
%          MD      : Matrix of differentially encoded complex values (unit magnitude)
%                    and one of N-ary possible phases (N-PSK)
%          B       : Matrix of 8-ary symbols
%          nsymno: Number of N-ary generated symbols
%
%Subroutines Used : marymsg.m,cdlilv.m,mb.m,bm.m,difcdrft.m,cmv2fa.m
%-------------------------------------------------------------------------------
function
[Fa,MD,B_ce,B_random,nsymno]=cdrcdlft(picy_n,pic,case,s,freqno,rintlv,cintlv,N,mary,
nary,fort);
%
%Determine whether the number of OFDM frequencies are even (# of matrix
columns),indicated
% by the "freqno" parameter.  If odd go to error message. Odd frequencies are not
allowed
% since the formation of the frequency array is symmetrical and even.
```

```
%
if rem(freqno,2)~=0
   disp('ERROR: The number of matrix columns,freqno,representing OFDM frequencies
must be an even number!')
elseif rem(freqno,2)==0
%
% Determine if the row and column interleave parameters are greater than freqno when
% multiplied together.  If not, then display error message and stop.
%
   if (rintlv*cintlv)<(freqno)
      disp('')
      disp('ERROR: The row and column interleave parameters are not compatible with #
of OFDM frequencies!')
      disp('')
   else
% Calculate the row symbol number
      symno=rintlv*cintlv/freqno;
%
% Display error message if symno and freqno not compatible with rintlv and cintlv and
stop.
% If not compatible,the interleaver function does not work correctly.
%
      if rem(symno,1)~=0
         disp('')
         disp('ERROR: The row and column interleave parameters are not compatible with #
of OFDM frequencies!')
         disp('  For the enetered rintlv, cintlv, and freqno parameters, the calculated symno
is:')
         disp(symno)
         multiesall=mltpl(rintlv,cintlv);
         multies=multiesall(1,(2:length(multiesall)-1));
         disp('  Possible choices for freqno based upon rintlv and cintlv are:')
         disp('')
         disp(multies)
      elseif rem(symno,1)==0
         if freqno >= N;
            disp('')
            disp('ERROR: The number of frequency points, N, needs to be incresaed !')
            disp('N must be larger than:')
            disp('')
            disp(freqno)
            disp('')
         elseif freqno < N;
            %
            % Generate a random message matrix of m-ary symbols,based upon
parameter,mary.
```

```
          %
          Nmbr_of_symbols =symno*freqno;
          %
          [B_ce,B_random]=marymsg(mary,s,symno,freqno);
          Rndm_m_ary_msg=B_random;
          %
          % Perform a block interleaving function on the matrix, B, with rintlv rows
          % and cintlv columns.
          %
          SYNC=[];
          [Br Bc]=size(B_ce);
          Bt=B_ce';
          Bvect=Bt(:)';
          si=cdlilv(rintlv,cintlv,case,Bvect,SYNC);
          Bi=reshape(si,Bc,Br)';
          Intrlvd_array=Bi;
          %
          m1=bm(nary,mb(mary,Bi));
          lengthm1=length(m1);
          nsymno=lengthm1;
          remm1=rem(lengthm1,freqno);
          if remm1==0;
            m1=m1;
          else
            zero=zeros(freqno-remm1);
            m1=[m1 zero(1,:)];
          end
          length2m1=length(m1);
          m=(reshape(m1,freqno,length2m1/freqno))';
          N_ary_msg=m;
          %
          % Generate a differentially encoded matrix of complex
          % values with unit magnitude and one of (2^n) equal phases.
          MDD=difcdrft(nary,m,fort);
          [MDm MDn]=size(MDD);
          MD=MDD;
          Cmplx_mod_array=MDD;
          %
          % Form the frequency array of modulation values that include guard interval.
          %
          Fa=cmv2fa(N,MD);
          Freq_array=Fa;
        end
      end
    end
  end
```

CHANCDL

```
%function [errmax,errors,freqerrs]=
chancdl(chnmdl,wait,prnt,picy_n,pic,case,s,freqno,rintlv,cintlv,N,mary
%nary,n,k,blklgth,N,g,sigs,loss,dly,dop,freqspace,fort)
%--------------------------------------------------------------------------------
%
%Title              : Simulations for AWGN & Multipath Fading Channel
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%--------------------------------------------------------------------------------
%Subroutines Used : cdrcdlft.m,tda.m,awgn.m,chuhf.m,itda.m,decdrcdl.m,check.m
%--------------------------------------------------------------------------------
%
function[errmax,errors,freqerrs]=chancdl(chnmdl,wait,prnt,picy_n,pic,case,s,freqno,rintlv
,cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,loss,dly,dop,freqspace,fort)
sigvect=sigs;
klgth=length(k);
chklp=1;
errvect=[];
bervect=[];
freqerrmx=[];
errsperpr=[];
Es_No=[];
Eb_No=[];
sermx=[];
bermx=[];
rowerrmx=[];
symno=rintlv*cintlv/freqno;
for lp=1:length(sigvect);
%
[xmt,modvals,B_ce,B_random,nsymno]=cdrcdlft(picy_n,pic,case,s,freqno,rintlv,cintlv,N,
mary,nary,fort);
xmtifft=tda(Ng,xmt);
xmtpts=1:length(xmtifft);
if chnmdl==0
  %
  sandn=xmtifft;
elseif chnmdl==1
  %
  disp(['Sigma=',num2str(sigvect(lp))]);
  sandn=awgn(xmtifft,s,N,sigvect(lp));
elseif chnmdl==2
  %
  sandn=chuhf(s+1,xmtifft,loss,dly,dop,N,freqspace);
```

```
elseif chnmdl==3
   %
   sandmltpth=chuhf(s+1,xmtifft,loss,dly,dop,N,freqspace);
   disp(['Sigma=',num2str(sigvect(lp))]);
   sandn=awgn(sandmltpth,s,N,sigvect(lp));
end
%
sandnfft=itda(Ng,sandn);
%
K=(length(modvals(1,:)))/2;
[rcvd,rcvd_bit,random_msg,random_bit,M,MM]=decdrcdl(picy_n,pic,case,K,sandnfft,ns
ymno,freqno,rintlv,cintlv,mary,nary,fort,B_random);
%Transmitted_msg=B_random;
Transmitted_msg=random_msg;
Received_msg=rcvd;
%
[errors,bit_error,freqerrs,errmx,rowerrs]=check(pic,random_msg,random_bit,rcvd,rcvd_b
it,n,k(chklp),blklgth);
errvect=[errvect,errors];
bervect=[bervect,bit_error];
freqerrmx=[freqerrmx;freqerrs];
rowerrmx=[rowerrmx;rowerrs];
crntEs_No=1/(2*N*(sigvect(lp)^2));
%
%based on M=4 i.e. for coded QPSK, Eb=Es.
crntEb_No=crntEs_No;
Es_No=[Es_No,crntEs_No];
Eb_No=[Eb_No,crntEb_No];
Es_Nodb=10*log10(Es_No);
Eb_Nodb=10*log10(Eb_No);
end
ser=errvect/(symno*freqno);
ber=bervect/(2*symno*freqno);
sermx=[sermx;ser];
bermx=[bermx;ber];
errsum=sum(errvect);
errsperpr=[errsperpr,errsum];
errmax=max(rowerrmx');
%
% plot
%
if picy_n==1
   figure(pic+1)
   plot(modvals,'*')
   hold on;
   plot(0,0,'+')
```

```
  hold off;
  title(['Transmitted Signal',int2str(2^nary),'-ary Constellation Plot'])
  xlabel(['Magnitude=1'])
  axis('square');
  orient tall
  grid
  if prnt==1;
    print
    pause(10);
  end
  pause(wait);
end
%
%
if picy_n==1
  figure(pic+2)
  plot([0:N-1],abs(xmt),'*')
  title(['Frequency Array Plot (number of FFT frequency points are ',int2str(N),')'])
  xlabel(['Guard interval length is ',int2str(N-freqno)])
  axis('square');
  orient tall
  grid
  if prnt==1;
    print
    pause(10)
  end
  pause(wait);
end
%
%
if picy_n==1
  figure(pic+3)
  surf(abs(modvals));
  shading interp
  grid
  orient tall
  title(['Magnitude of Transmitted Signal(Unity Magnitude)'])
  xlabel('OFDM Freq #')
  ylabel('Symbol Row Number')
  zlabel(['Magnitude(seed =',int2str(s),')'])
  if prnt==1;
    print
    pause(10)
  end
  pause(wait);
end
```

```
%
%
if picy_n==1
   figure(pic+6)
   plot(M,'*')
   hold on;
   plot(0,0,'+')
   hold off;
   title(['Received',int2str(2^nary),'-ary Signal Constellation Plot,before Differential
Decoding'])
   orient tall
   axis('square');
   grid
   if prnt==1;
      print
      pause(10)
   end
   pause(wait);
end
%
%
if picy_n==1
   figure(pic+7)
   plot(MM,'+')
   hold on;
   plot(0,0,'+')
   hold off;
   title(['Received',int2str(2^nary),'-ary Sigal Constellation Plot, After Differential
Decoding'])
   orient tall
   axis('square');
   grid
   if prnt==1;
      print
      pause(10)
   end
   pause(wait);
end
%
%
if picy_n==1
   roty_n=input('Do you want to rotate 3-D plot?(yes=1,no=0):');
   figure(pic+8)
   surf(abs(M));
   shading interp
   grid
```

```
  orient tall
  title(['Magnitude Variation of Received Signal (Sigma=',num2str(sigvect(lp)),')'])
  xlabel('OFDM Freq #')
  ylabel('Symbol Row Number')
  zlabel(['Magnitude(seed=',int2str(s),')'])
  if roty_n==1
    %Rotate the 3-D plot
    for k=1:5
      view(-70+10*k,15+10*k)
      disp('');
      disp('Press "enter" to rotate plot...');
      pause
    end
  end
  if prnt==1;
    print
    pause(10)
  end
  pause(wait);
end
%
  if errsum~=0
    %
    %2-D Error Performance Curve showing BER vs. Es/No.
    %
    figure(pic+12)
    semilogy(Eb_Nodb,ber)
    grid
    if fort==1
      if dop==[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
        title(['Link 1 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
      elseif dop==[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
        title(['Link 2 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
      elseif dop==[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15]
        title(['Link 3 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
      else
        title(['Custom Link Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
      end
    elseif fort==0
      if dop==[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
        title(['Link 1 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
```

```
    elseif dop==[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
      title(['Link 2 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
    elseif dop==[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15]
      title(['Link 3 : Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
    else
      title(['Custom Link Performance graph: BIT Error Rate vs. Eb/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
    end
  end
  %
  xlabel(['Eb/No(dB)(# of OFDM=',int2str(freqno),')(case=',int2str(case),')(Interleaver
pair=',int2str(rintlv),',',int2str(cintlv),') M-ary=',int2str(2^mary),',N-
ary=',int2str(2^nary)]);
  ylabel(['Sigma Range:(',num2str(min(sigs)),'-',num2str(max(sigs)),')(R-
S=',int2str(floor((n-k)/2)),')(Symbol#=',int2str(symno*freqno),')(Seed=',num2str(s),')']);
  orient tall
  %
  %2-D Error Performance Curve showing SER vs. Eb/No.
  %
  figure(pic+13)
  semilogy(Es_Nodb,ser)
  grid
  if fort==1
    if dop==[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
      title(['Link 1 : Performance graph: Symbol Error Rate vs. Es/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
    elseif dop==[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
      title(['Link 2 : Performance graph: Symbol Error Rate vs. Es/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
    elseif dop==[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15]
      title(['Link 3 : Performance graph: Symbol Error Rate vs. Es/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
    else
      title(['Custom Link Performance graph: Symbol Error Rate vs. Es/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
    end
  elseif fort==0
    if dop==[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
      title(['Link 1 : Performance graph: Symbol Error Rate vs. Es/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
    elseif dop==[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
      title(['Link 2 : Performance graph: Symbol Error Rate vs. Es/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
    elseif dop==[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15]
```

```
        title(['Link 3 : Performance graph: Symbol Error Rate vs. Es/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
      else
        title(['Custom Link Performance graph: Symbol Error Rate vs. Es/No (Freq. Diff.
Enc.)(Total errors=',int2str(sum(errvect)),')'])
      end
    end
    text(min(ceil(Es_Nodb)),.18,['Loss=[',num2str(loss),']']);
    text(min(ceil(Es_Nodb)),.12,['Delay=[',num2str(dly),']']);
    text(min(ceil(Es_Nodb)),.08,['Doppler=[',num2str(dop),']']);
    xlabel(['Es/No(dB)(# of OFDM=',int2str(freqno),')(case=',int2str(case),')(Interleaver
pair=',int2str(rintlv),',',int2str(cintlv),') M-ary=',int2str(2^mary),',N-
ary=',int2str(2^nary)]);
    ylabel(['Sigma Range:(',num2str(min(sigs)),'-',num2str(max(sigs)),')(R-
S=',int2str(floor((n-k)/2)),')(Symbol#=',int2str(symno*freqno),')(Seed=',num2str(s),')']);
    orient tall
    end
  if prnt==1
    print
    pause(10)
  end
  %
```

CHECK

```
%function[error_no,bit_error_total,freqerrs,errmx,rowerrs]=check(pic,x,y,n,k,blklgth)
%
%--------------------------------------------------------------------------------
%
%Title              : Source and Sink Message Checker
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Prof. Paul H. Moose, Naval Postgraduate School
%Modified by        : Tan Kok Chye, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
%
function[error_no,bit_error_total,freqerrs,errmx,rowerrs]=check(pic,x,xbit,y,ybit,n,k,blkl
gth)
if blklgth>n
  disp('')
  disp('ERROR! The block length, blklgth, must be equal or less than the code word
length,n.')
  disp('Please enter a smaller value for blklgth, or change n.')
  disp('')
elseif blklgth<=n
  if n<k
    disp('')
    disp('Error! The code word length,n,must be equal or larger than the information
length,k.')
    disp('Please enter a larger value for n, or chnage k to a smaller number.')
    disp('')
  elseif n>=k
    First_matrix=x;
    Second_matrix=y;
    [rx cx]=size(x);
    %
    %Compare inputs x and y and generate error matrix, "errors"
    %
    errors=(x~=y);
    %
    First=xbit;
    Second=ybit;
    [rx1 cx1]=size(xbit);
    %
    %Compare inputs xbit and ybit and generate BIT error matrix, "bit_errors"
    %
    bit_errors=(xbit~=ybit);
    %
    %Find the error distribution vs. OFDM frequencies
```

```
%
freqerrs=sum(errors);
%
%Find the error location in "errors" where element in x and y differ.
%
Error_locations=(find(errors))';
Error_number=sum(sum(errors));
Correct_symbl_num=(size(y,1)*size(y,2))-Error_number;
%
%Find the bit error location in "errors" where element in x1 and y1 differ.
%bit_Error_locations=(find(bit_errors))';
bit_error_total=sum(sum(bit_errors));
%Correct_bit_num=(size(y1,1)*size(y1,2))-bit_Error_number;
%
%
%Reed-Solomon 8-bit symbol correction for (n-k)/2 symbols
%
symcorr=floor((n-k)/2);
if blklgth<=(n-k)
  disp('Error!!!The block length is too short for the given n and k values')
  disp('')
elseif blklgth>(n-k)
  errtrans=errors';
  %
  %Reshape the error matrix as a vector of errors
  %
  errvect=errtrans(:)';
  %
  blkrem=rem(length(errvect),blklgth);
  if blkrem~=0;
    zeropad=zeros(blklgth-blkrem);
    errvectpad=[errvect zeropad(1,:)];
  elseif blkrem==0;
    errvectpad=errvect;
  end
  %
  blknos=length(errvectpad)/blklgth;
  %
  errcorct=[];
  errblksum=[];
  %
  for lp=1:blknos;
  errblk=errvectpad(((blklgth*(lp-1))+1):(blklgth*lp));
  errblklgth=length(errblk);
  if sum(errblk)<=symcorr;
    noerr=zeros(errblklgth);
```

```
        errblk=noerr(1,:);
      elseif sum(errblk)>symcorr;
        errblk=errblk;
      end
      errcorct=[errcorct errblk];
      errblksum=[errblksum sum(errblk)];
    end
    newerrvect=errcorct(1:length(errvect));
    errtot=sum(newerrvect);
    RSerrs=(reshape(newerrvect,size(errors,2),size(errors,1)))';
    %
    %Find the error distribution vs. OFDM Frequencies
    %
    freqerrs=sum(RSerrs);
    errindex=(find(RSerrs))';
    RSerrtot=sum(errblksum);
    RSerrdif=Error_number-RSerrtot;
    errperblk=[(1:blknos);errblksum];
    %
    %Check to see if x and y are the same. If not, display error message
    %
    if x==y;
      disp('GREAT!!!there are no errors.')
      error_no=0;
      errmx=errors;
      rowerrs=sum(errors');
    else
      disp('WARNING!:Errors were detected!')
      disp('')
      if n==k
      disp('WARNING!: Since n=k,there is no R-S error correcting possible')
      disp('')
    end
    disp(['For the given input parameters:n=',int2str(n),'and k=',int2str(k),',the Reed-
Solomon code is capable'])
    disp(['of correcting',int2str(symcorr),'errors.'])
    disp('')
    %
    %Check to see if xbit and ybit are the same. If not, display error message
    %
    if xbit==ybit;
      disp('GREAT!!!there are no bit errors.')
      bit_error_no=0;
      bit_errmx=bit_errors;
      bit_rowerrs=sum(bit_errors');
    else
```

```
      disp('WARNING!:Errors were detected!')
      disp('')
      if n==k
      disp('WARNING!: Since n=k,there is no R-S error correcting possible')
      disp('')
      end
    end
    %
    %RS code was able to correct all errors
    %
    if errtot==0
      Pre_RS_error_matrix=errors;
      disp('EXCELLENT: The Reed-Solomon code corrected all detected errors!')
      disp(['Originally the error total was:',int2str(Error_number)])
      disp('')
      error_no=0;
      errmx=zeros(rx,cx);
      rowerrs=sum(errmx');
      %
      %RS code was able  to correct some errors but not all of them
      %
    elseif errtot<Error_number
      Pre_RS_error_matrix=errors;
      Post_RS_error_matrix=RSerrs;
      errmx=RSerrs;
      rowerrs=sum(errmx');
      disp('OOOPS: The Reed-Solomon code corrected some detected errors, but not all.')
      disp(['Originally the error total was : ',int2str(Error_number)])
      disp('')
      disp(['After R-S decoding , the error number was reduced to:',int2str(RSerrtot)])
      disp('')
      error_no=RSerrtot;
      disp(['The total number of correct symbols are:',int2str((size(y,1)*size(y,2))-
RSerrtot)])
    disp('')
    disp('The error number distribution per block number is :')
    disp(errperblk)
    %figure(pic+3)
    %bar((1:blknos),errblksum)
    %axis([0.5(blknos+0.5)0(max(errblksum)+1)])
    %title(['Simulation#',int2str(pic),':Error Distribution Per Message Block (Error
count=',int2str(error_no),')'])
    %xlabel(['Message Block Number(block size:',int2str(blklgth),'symbols)'])
    %
    %RS code did not correct any errors
    %
```

```
  elseif errtot==Error_number
    Error_matrix=errors;
    errmx=errors;
    rowerrs=sum(errors');
    disp('OOOPS!:The Reed-Solomon code did not correct any errors.')
    disp('Perhaps a more powerful R-S code is required.')
    disp('')
    disp(['The total number of error occurrences is:',int2str(Error_number)])
    disp('')
    error_no=errtot;
    disp('The error number distribution per block number is :')
    disp(errperblk)
    %figure(pic+4)
    %bar((1:blknos),errblksum)
    %axis([0.5 (blknos+.5) 0 (max(errblksum)+1)])
    %title(['Simulation#',int2str(pic),':Error Distribution Per Message Block. (Error count
=',int2str(error_no),')'])
    %xlabel(['Message Block Number (block size:',int2str(blklgth)',symbols)'])
    %
   end
end
end
end
end
disp('_____________________________________________________');
```

CHN0CDL

```
%-------------------------------------------------------------------------------
%
%Title              : Model Zero (Noise Free) simulation
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function
[errmax,errors,freqerrs]=chn0cdl(prnt,picy_n,pic,case,s,freqno,rintlv,cintlv,N,mary,nary,
n,k,blklgth,Ng,fort)
disp(' ');
klgth=length(k);
chklp=1;
errvect=[];
freqerrmx=[];
errsperpr=[];
Es_No=[];
sermx=[];
rowerrmx=[];
symno=rintlv*cintlv/freqno;
%
[xmt,modvals,B,nsymno]=cdrcdlft(picy_n,pic,case,s,freqno,rintlv,cintlv,N,mary,nary,fort
);
Random_Source_Msg=B
%
xmtifft=tda(Ng,xmt);
xmtpts=1:length(xmtifft);
%
sandnfft=itda(Ng,xmtifft);
K=(length(modvals(1,:)))/2;
[rcvd,M]=decdrcdl(picy_n,pic,case,K,sandnfft,nsymno,freqno,rintlv,cintlv,mary,nary,fort
);
Transmitted_msg=B;
Sink_msg=rcvd
%
%
[errors,freqerrs,errmx,rowerrs]=check(pic,B,rcvd,n,k(chklp),blklgth);
errvect=[errvect,errors];
freqerrmx=[freqerrmx;freqerrs];
rowerrmx=[rowerrmx;rowerrs];
end
ser=errvect/(symno*freqno);
```

```
sermx=[sermx;ser];
errsum=sum(errvect);
errsperpr=[errsperpr,errsum];
errmax=max(rowerrmx');
if errsum==0;
   disp('Test Passed!!!')
   disp('')
elseif errsum~=0;
   disp('WARNING! Test Failed!')
   disp('')
end
%
```

CHUHF

```
%--------------------------------------------------------------------------------
%
%Title                 : UHF Channel Model (multipath Channel Model2)
%Thesis Advisor        : Prof J. McEachen, Naval Postgraduate School
%Author                : Prof. Paul H. Moose, Naval Postgraduate School
%Modified       by     : Tan Kok Chye, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
%
function y=chuhf(s,x,loss,dly,dop,N,freqspace)
c=10.^(-loss./20);
deltat=1/(N*freqspace);
d=(dly.*.000001)./deltat;
e=dop./freqspace;
[L,Nt]=size(x);
D=length(d);
x=x.';
x=x(:).';
%
%D path with delays from d. (Uses macro dline.m)
%
xd=dline(x,d);
[rr,cc]=size(xd);
x=xd(1,:);
% Offsets direct path by .7 of max doppler freq. (uses macro ofst.m)
%
xo=ofst(.7*e(1),N,x);
%
% First path with no fading. (uses macro ray_dop.m)
%
for l=1:D
   a=ray_dop(s,cc,N,e(1));
   xd(1,:)=a.*xd(1,:);
end
%Sums the fading paths
y=c*xd;
%
%Adds in the First path without fading for the GSM-Ricean.
%
y=y+xo;
y=y(1:L*Nt);
y=reshape(y,Nt,L).';
%_____________________________________________________________________
```

CMV2FA

```
%-------------------------------------------------------------------------------
%
%Title              : Complex Frequency Array Generator
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function X=cmv2fa(N,M)
[m n]=size(M);
if rem(n,2)==0;
  M=M;
else
  %
  M=[zeros(m,1) M];
end
[m n]=size(M);
K=round(n/2);
%
%Generate a matrix of zeros with m row and N columns.
%
X=zeros(m,N);
%
X(:,1:K)=M(:,K+1:2*K);
X(:,N-K+1:N)=M(:,1:K);
%_____________________________________________________
```

CMVDIFCK

```
%-------------------------------------------------------------------------------
%
%Title              : Frequency Array & Differential Encoder/Decoder Verifier
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : DaveRoderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function cmvdifck(s,symno,freqno,N,mary,nary)
disp('_____________________________________________________')
disp('This m-file checks the correctness of the differential encloder/decoder & the
frequnecy arrangers.')
fort=input('To run the frequency version, enter 1(one); otherwise, enter 0(zero) torun the
time version:');
%
%Generate random m-ary message array
%
B=marymsg(mary,s,symno,freqno);
Source_msg=B
[Br Bc]=size(B);
Bt=B';
Bvect=Bt(:)';
si=Bvect;
Bi=reshape(si,Bc,Br)';
%
%
%
m1=bm(nary,mb(mary,Bi));
lengthm1=length(m1);
nsymno=lengthm1;
remm1=rem(lengthm1,freqno);
if remm1==0;
  m1=m1;
else
  zero=zeros(freqno-remm1);
  m1=[m1 zero(1,:)];
end
length2m1=length(m1);
m=(reshape(m1,freqno,length2m1/freqno))';
N_ary_msg=m;
%
% Generate a differentially encoded matrix of complex values with unit
% magnitude and one of (2^n) equal phases.
```

```
%
MDD=difcdrft(nary,m,fort);
[MDm MDn]=size(MDD);
MD=MDD;
Cmplx_mod_array=MDD;
%
% Form the frequency array of modulation values that include guard interval.
%
Fa=cmv2fa(N,MD);
Freq_array=Fa;
%
% Generate the corresponding complex modulation values from the received frequency
array.
%
K=(length(MD(1,:)))/2;
M=fa2cma(K,Fa);
Cmplx_mod_vals=M;
%
% Perform differential decoding.
%
naryp=nary;
[s,MM]=dfdcdrft(naryp,nary,M,fort);
[L,cc]=size(s);
strans=s';
svect=strans(:)';
corrs=svect(1:nsymno);
%
% Convert from N-ary symbols to the final message format of M-ary symbols.
%
nsymno;
Br=bm(mary,mb(nary,corrs));
lengthBr=length(Br);
rmndr=rem(length(Br),freqno);
if rmndr==0;
   Br=Br;
elseif rmndr~=0;
   Br=Br(1:(lengthBr-rmndr));
end
rcvd=(reshape(Br,freqno,length(Br)/freqno))';
[Br Bc]=size(rcvd);
SYNC=[];
sr=rcvd';
si=sr(:)';
sd=si;
outmsg=reshape(sd,Bc,Br)';
Sink_Msg=outmsg
```

```
%
% Check results for correctness
%
[error_no,freqerrs,errmx,rowerrs]=check(0,B,rcvd,freqno,freqno,freqno);
if sum(rowerrs)==0
   disp('Test Passed!!!');
elseif sum(rowerrs)~=0
   disp('OOOOPS - Test Failed!')
end
disp('_____________________________________________________________')
%_____________________________
```

CNV_ENCD

```
%-------------------------------------------------------------------------------
%
%Title                 : Convoultional Encoding
%Thesis Advisor    : Prof J. McEachen, Naval Postgraduate School
% Reference         : Contemporary Communication System using MatLab
%John G. Proakis & Masoud Salehi.
%-------------------------------------------------------------------------------
%
function ce_output=cnv_encd(ce_g,ce_k0,ce_input)
% cnv_encd(ce_g,ce_k0,ce_input)
% determines the output sequence of a binary convolutional encoder
% ce_g is the generator matrix of the convolutional code
% with ce_n0 rows and ce_l*ce_k0 columns.  Its rows are ce_g1,ce_g2,....ce_gn.
% ce_k0 is the number of bits entering the encoder at each clock cycle.

% check to see if extra zero padding is necessary
if rem(length(ce_input),ce_k0)>0
   ce_input=[ce_input,zeros(size(1:ce_k0-rem(length(ce_input),ce_k0)))];
end
ce_n=length(ce_input)/ce_k0;
%check the size of matrix ce_g
if rem(size(ce_g,2),ce_k0)>0
   error('Error, ce_g is not of the right size.')
end
% determine ce_l and ce_n0
ce_l=size(ce_g,2)/ce_k0;
%disp(['The value of ce_l is:',int2str(ce_l)]);
ce_n0=size(ce_g,1);
%disp(' ')
%disp(['The value of ce_n0 is:',int2str(ce_n0)]);
%add extra zeros
ce_u=[zeros(size(1:(ce_l-1)*ce_k0)),ce_input,zeros(size(1:(ce_l-1)*ce_k0))];
%generate ce_uu, a matrix whose column are the contents of
%conv. encoder at various cycles.
ce_u1=ce_u(ce_l*ce_k0:-1:1);
for ce_i=1:ce_n+ce_l-2
   ce_u1=[ce_u1,ce_u((ce_i+ce_l)*ce_k0:-1:ce_i*ce_k0+1)];
end
ce_uu=reshape(ce_u1,ce_l*ce_k0,ce_n+ce_l-1);
%determine the ce_output
ce_output=reshape(rem(ce_g*ce_uu,2),1,ce_n0*(ce_l+ce_n-1));
```

CODERIFT

```
%--------------------------------------------------------------------------------
%
%Title              : COFDM Encoder Without Interleaving
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
function [Fa,MD,B,nsymno]=coderift(picy_n,pic,s,freqno,rintlv,cintlv,N,mary,nary,fort);
%
if rem(freqno,2)~=0
  disp('ERROR: The number of matrix column, freqno, representing OFDM frequencies
must be an even number!')
elseif rem(freqno,2)==0
  %
  if(rintlv*cintlv)<(freqno)
    disp('')
    disp('ERROR: The row and column interleave parameters are not compatible with #
of OFDM frequencies!')
    disp('')
  else
    %
    symno=rintlv*cintlv/freqno;
    %
    if freqno>=N;
      disp('')
      disp('ERROR: The number of frequency points, N, needs to be increased!')
      disp('N must be larger than:')
      disp('')
      disp(freqno)
      disp('')
    elseif freqno<N;
      Nmbr_of_symbols=symno*freqno;
      %
      B=marymsg(mary,s,symno,freqno);
      Rndm_m_ary_msg=B;
      %
      m1=bm(nary,mb(mary,B));
      lengthm1=length(m1);
      nsymno=lengthm1;
      remm1=rem(lengthm1,freqno);
      if remm1==0;
        m1=m1;
```

```
        else
          zero=zeros(freqno-remm1);
          m1=[m1 zero(1,:)];
        end
        length2m1=length(m1);
        m=(reshape(m1,freqno,length2m1/freqno))';
        N_ary_msg=m;
        %
        MDD=difcdrft(nary,m,fort);
        [MDm MDn]=size(MDD);
        MD=MDD;
        Cmplx_mod_array=MDD;
        %
        Fa=cmv2fa(N,MD);
        Freq_array=Fa;
        %
        if picy_n==1
          if pic==1
            figure(pic)
            plot(MD,'*')
            hold on;
            plot(0,0,'+')
            hold off;
            title([Transmitted Signal',int2str(nary),'-ary Constellation Plot'])
            axis('square');
            orient tall
            grid
            %
            figure(pic+1)
            %
            xaxis=[0:N-1];
            plot(xaxis,abs(Fa),'*')
            title(['Frequency Arrary Plot (number of frequency point are',int2str(N),')'])
            xlabel(['Guard interval length is',int2str(N-freqno)])
            axis('square');
            orient tall
            grid
          end
        end
      end
    end
  end
end
end
end
end
%_____________________________________________________________________
```

COFDMSIM

```
%-------------------------------------------------------------------------------
%
%Title              : Simulation Of COFDM
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
disp('_____________________________________________________________________');
disp('This batch m-file runs COFDM simulations using different channel models.')
fort=input('To run the frequency version, enter 1(one), To run the time version, enter 0
(zero), or to run both enter 2(two):');
freqno=input('Enter the # of OFDM frequencies (note : must be even):');
N=input('Enter the number of FFT points (Note : This number must be larger than # of
OFDM frequencies):');
chnmdl=input('Do you want to run channel model 0, channel model 1, channel model 2
or channel model 3 ? (Enter 0,1,2 or 3):');
if chnmdl==0
   disp('Channel model 0 simulation performed.');
   sigs=0;
   loss=0;
   dop=0;
   dly=0;
elseif chnmdl==1
   disp('Channel model 1 simulation performed.');
   sigs=input('Enter the sigma noise parameter range or single value. (Ex
linspace(0,0.02,20)or .003):');
   loss=0;
   dop=0;
   dly=0;
elseif chnmdl==2
   disp('Channel model 2 simulation performed.');
   sigs=0;
   pthno=input('Do you want to run link1 ,link2, link3 or a custom link ? (Enter 1,2,3 or 4
for custom):');
   %
   %Link parameters
   %
   if pthno==3
      %my link 3

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
```

```
    dop=[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

  elseif pthno==2
    %my link 2

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
    dop=[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

  elseif pthno==1
    %my link 1

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
    dop=[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

  elseif pthno==4
    disp('Custom link simulation...')
    loss=input('Enter the apth loss in db (Ex [0 4 7]):');
    dop=input('Enter the doppler frequency in Hertz (Ex [30 20 15]):');
    dly=input('Enter the time delays of the multipaths in microsecs (Ex [0 0.6 3.9]):');
  end
elseif chnmdl==3
  disp('Channel model 3 simulation performed');
  sigs=input('Enter the sigma noise parameter range or single value. (Ex linspace
(0,0.02,20) or .003):');
  pthno=input('Do you want to run link1, link2, link3 or a custom link ?(Enter 1,2,3 or 4
for custom):');
  %
  %Link parameters
  %
  if pthno==3
    %my link 3

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
    dop=[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15];
```

```
dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

  elseif pthno==2
    %my link 2

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
    dop=[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

  elseif pthno==1
    %my link 1

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
    dop=[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

  elseif pthno==4
    disp('Custom link simulation...')
    loss=input('Enter the apth loss in db (Ex [0 4 7]):');
    dop=input('Enter the doppler frequency in Hertz (Ex [30 20 15]):');
    dly=input('Enter the time delays of the multipaths in microsecs (Ex [0 0.6 3.9]):');
  end
end
allcase=input('Simulate all interleaver cases (yes) or specific ones(no)? (1=yes,0=no):');
if allcase==1
  disp('All cases,(0-8),will be tested.');
  cases=[0:8];
elseif allcase==0
  cases=input('Enter specific case numbers from (0 to 8)(Ex [0 4 5 8]):');
end
if fort~=2
  if length(cases)~=1
    casey_n=input('Do you want to find optimal interleaver case(s) ? (1=yes, 0=no):');
  end
end
totsym=input('Enter the total minimum number of symbols to simulate (Ex 10000):');
rowno=ceil(totsym/freqno);
if totsym~=(rowno*freqno)
```

```
   disp(['Note:Based on the parameters thus far, the actual total number of symbol to be
simulated will be :',int2str(rowno*freqno)]);
end
pry_n=input('For the interleaver, do you want to calculate all possible intermediate
matrix dimension pairs?(1=yes,0=no):');
pair1=1;
pair2=rowno*freqno;
if pry_n==1
   %
   %
   %
   Intrlvr_pairs=intlvprs(rowno,freqno);
   intlvrprs=Intrlvr_pairs;
   disp('')
   disp('For these input parameters, all possible acceptable interleaver dimension pairs are
: ')
   disp(Intrlvr_pairs)
end
pairs=input(['Desired interleaver pair? (Ex [row # col #] = [20 50] (Note: entering
[',int2str(pair1),' ',int2str(pair2),'],or [',int2str(pair2),' ',int2str(pair1),'], offers no
interleaving functionality):']);
rintlv=pairs(1);
cintlv=pairs(2);
mary=input('Enter the number of M-ary bits, q (i.e. for 256-ary, q=8):');
nary=input('Enter the number of N-ary bits,q(i.e. for 16-ary, q=4):');
freqspace=round(16600000/freqno);
Ng=input('Enter the guard interval length (Number of sample points):');
ecc=input('Do you want to include error correction coding ? (1=yes, 0=no):');
if ecc==1
   code=input('Entern,k and error correction block length (Ex [240 200 240]):');
   n=code(1);
   k=code(2);
   blklgth=code(3);
elseif ecc==0
   n=freqno;
   k=freqno;
   blklgth=freqno;
end
svals=input('Enter specific seed values, or 0 for a random seed (ex [103 22, 60] or [0]):');
picy_n=input('Do you want signal plots? (1=yes, 0=no):');
if picy_n==1
   wait=input('How many seconds of delay between pictures?');
   wait=round(wait);
elseif picy_n==0
   wait=0;
end
```

```
prnty_n=input('Do you want print outs? (1=yes, 0=no):');
pic=0;
svect=[];
for run=1:length(svals);
  errcase=[];
  errtot=[];
  if min(svals)==0
    rand('seed',sum(100*clock));
    s=round(abs(rand(1)*pi*10*(pic+1)*run));
  elseif min(svals)~=0
    s=svals(run);
  end
  svect=[svect,s];
  for l=1:length(cases);
    disp('_____')
    disp(['Run#:',int2str(run)]);
    disp(['Seed=',int2str(s)]);
    disp(['Interleaver case=',int2str(cases(l))]);
    %
    %
    if fort<=1

[errmax,errors,freqerrs]=chancdl(chnmdl,wait,prnty_n,picy_n,pic,cases(l),s,freqno,rintlv,
cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,loss,dly,dop,freqspace,fort);
    elseif fort==2
      disp('Frequency differential encoding/decoding simulation...')
      disp('_____')

[errmax,errors,freqerrs]=chancdl(chnmdl,wait,prnty_n,picy_n,pic,cases(l),s,freqno,rintlv,
cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,loss,dly,dop,freqspace,1);
      disp('**********************************************************')
      disp('Time differential encoding/decoding simulation....')
      disp('_____')

[errmax,errors,freqerrs]=chancdl(chnmdl,wait,prnty_n,picy_n,pic+12,cases(l),s,freqno,rin
tlv,cintlv,N,mary,nary,n,k,blklgth,Ng,sigs,loss,dly,dop,freqspace,0);
    end
    errtot=[errtot sum(errors)];
    errvect=[errvect,errtot];
    errcase=[errcase sum(errmax)];
  end
  if fort~=2
    casearry=[cases;errcase];
    %
    %
    %
```

```
      %if casey_n==1
       %  figure(pic+13)
        % bar(cases,errcase)
         %grid
         %orient tall
         if fort==1
            title([int2str(pic),':Maximum Error Total Vs. Interleaver Case Number (Freq.
Diff. Enc.) (OFDM Freq.#=',int2str(freqno),')'])
         elseif fort==0
            title([int2str(pic),':Maximum Error Total Vs. Interleaver Case Number (Time
Diff. Enc.) (OFDM Freq.#=',int2str(freqno),')'])
         end
         xlabel(['CDL Interleaver Case Number'])
         ylabel(['Maximum Error Count For Any Symbol Row (Seed=',int2str(s),')'])
         axis([-.5 8.5 0 (max(errcase)+1)])
         if prnty_n==1;
            print
            pause(10)
         end
         pause(wait);
         %
         figure(pic+14)
         bar(cases,errtot)
         grid
         orient tall
         title([int2str(pic),':Error Totals Vs. Interleaver Case Number'])
         xlabel(['CDL Interleaver Case Number'])
         ylabel(['Sigma:(',num2str(min(sigs)),'-',num2str(max(sigs)),') Error Total'])
         axis([-.5 8.5 (min(errtot)-1) (max(errtot)+1)])
         if prnty_n==1;
            print
            pause(10)
         end
         pause(wait);
      end
      pic=pic+1;
   end
end
disp('**********************************************************************'
)
disp('')
disp('Channel model batch run is finished!')
Seed=svect
%_____________________________________________________________________
```

CVDD

```
%-------------------------------------------------------------------------------
%
%Title                  : Continuous Variable digital delay element.
%Thesis Advisor    : Prof J. McEachen, Naval Postgraduate School
%Reference          : C.W. Farrow, " A Continuously Variable Digital Element", IEEE
%International Symposium on Circuits & Systems,pp.2641-2645,1988.
%Modified       by  : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function [y]=cvdd(x,alpha)
if ((nargin~=2)|(nargout~=1))
   error('ERROR:usage:y=y=cvdd(x,alpha);');
   return;
end
if (size(x)~=size(alpha))
   error('ERROR:x and alpha must be the same size');
   return;
end
if (abs(alpha)>0.5)
   error('ERROR:alpha must be within -0.5 and 0.5');
   return;
end
%
%-----------------------------------------------------
% Initialization
%-----------------------------------------------------
%
% Initialize FIR filter coefficients are in [1] (0,0.328 pass band)
C0=[-0.013824 0.054062 -0.157959 0.616394 0.616394 -0.157959 0.054062 -0.013824];
C1=[0.003143 -0.019287 0.1008 -1.226364 1.226364 -0.1008 0.019287 -0.003143];
C2=[0.055298 -0.216248 0.631836 -0.465576 -0.465576 0.631836 -0.216248 0.055298];
C3=[-0.012573 0.077148 -0.403198 0.905457 -0.905457 0.403198 -0.077148 0.012573];
%
%-----------------------------------------------------------
% 4 parallel FIR and add together based on [1]
%-----------------------------------------------------------
y0=filter(C0,[1],x);
y1=filter(C1,[1],x);
y2=filter(C2,[1],x);
y3=filter(C3,[1],x);
%
y=alpha.*y3;
y=alpha.*(y+y2);
```

```
y=alpha.*(y+y1);
y=y+y0;
%------------------------------------------------------
```

DECDRCDL

```
%-------------------------------------------------------------------------------
%
%Title              : COFDM Decoder With Deinterleaveing
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function[outmsg,viterbi_output_bit,random_msg,random_bit,M,MM]=decdrcdl(picy_n,p
ic,case,K,Fa,nsymno,freqno,rdintlv,cdintlv,mary,nary,fort,B_random)
%
%modified on 4 Jan.
%to generate BER vs Eb/No.
%to provide bit outputs
%
M=fa2cma(K,Fa);
Cmplx_mod_vals=M;
%
naryp=nary;
[s,MM]=dfdcdrft(naryp,nary,M,fort);
[L,cc]=size(s);
strans=s';
svect=strans(:)';
corrs=svect(1:nsymno);
%
%
nsymno;
Br=bm(mary,mb(nary,corrs));
lengthBr=length(Br);
rmndr=rem(length(Br),freqno);
if rmndr==0;
   Br=Br;
elseif rmndr~=0;
   Br=Br(1:(lengthBr-rmndr));
end
rcvd=(reshape(Br,freqno,length(Br)/freqno))';
Rcvd_Intlv_Ary=rcvd;
%
%
[Br Bc]=size(rcvd);
SYNC=[];
sr=rcvd';
si=sr(:)';
```

```
sd=cdldlv(rdintlv,cdintlv,case,si,SYNC);
viter_G=[1 0 1 1 0 1 1;1 1 1 1 0 0 1];
viter_k=1;
binary_value=mb(mary,sd);
[viterbi_output,survivor_sta,cumul_metrix]=viterbi(viter_G,viter_k,binary_value);
mary_dec=bm(mary,viterbi_output);
viterbi_output_bit=viterbi_output;
%outmsg=reshape(sd,Bc,Br)';
%
random_bit=B_random;
random_msg=bm(mary,random_bit);
[Brow Bcol]=size(random_msg);
%
outmsg=reshape(mary_dec,Bcol,Brow)';
Sink_Msg=outmsg;
%_____________________________________________________________________________
```

DECDRIFT

```
%-------------------------------------------------------------------------------
%
%Title              : COFDM Decoder Without Deinterleaving
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function
[outmsg]=decdrift(picy_n,pic,K,Fa,nsymno,freqno,rdintlv,cdintlv,mary,nary,fort)
M=fa2cma(K,Fa);
Cmplx_mod_vals=M;
%
if picy_n==1
   figure(pic+5)
   plot(M,'*')
   hold on;
   plot(0,0,'+')
   hold off;
   title(['Received',int2str(nary),'-ary Signal Constellation Plot, before Differential
Decoding'])
   orient tall
   axis('square');
   grid
end
%
naryp=nary;
[s,MM]=dfdcdrft(naryp,nary,M,fort);
[L,cc]=size(s);
strans=s';
svect=strans(:)';
corrs=svect(1:nsymno);
%
if picy_n==1
   figure(pic+6)
   plot(MM,'+')
   hold on;
   plot(0,0,'+')
   hold off;
   title(['Received',int2str(nary),'-ary Signal Constellation Plot, After Differential
Decoding'])
   orient tall
   axis('square');
```

```
   grid
end
%
nsymno;
Br=bm(mary,mb(nary,corrs));
lengthBr=length(Br);
rmndr=rem(length(Br),freqno);
if rmndr==0;
   Br=Br;
elseif rmndr~=0;
   Br=Br(1:(lengthBr-rmndr));
end
rcvd=(reshape(Br,freqno,length(Br)/freqno))';
M_ary_rcvd=rcvd;
outmsg=rcvd;
%_____________________________________________________________________________
```

DECI2BIN

```
%-------------------------------------------------------------------------------
%
%Title              : Convert Decimal To Binary
%Thesis Advisor    : Prof J. McEachen, Naval Postgraduate School
%Author             : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function y=deci2bin(x,l)
y=zeros(1,l);
vi=1;
while x>=0 & vi<=l
   y(vi)=rem(x,2);
   x=(x-y(vi))/2;
   vi=vi+1;
end
y=y(l:-1:1);
```

DFDCDRFT

```
%-------------------------------------------------------------------------------
%
%Title             : Complex Number Demodulator & Frequency/Time Differential Decoder
%Thesis Advisor    : Prof J. McEachen, Naval Postgraduate School
%Author            : Prof. Paul H. Moose, Naval Postgraduate School
%Modified      by  : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function [s,M]=dfdcdrft(qp,q,MD,fort)
if fort==0 %Time Differential decoding
  %
  %
  MD=MD';
  [m n]=size(MD);
  %
  % Perform a looping routine to find the phase differences between adjacent values in
the
  % array,MD,and put these calculated values into array,M.
  %
  for l=1:m
    for j=1:n-1
      M(l,j)=MD(l,j+1)*conj(MD(l,j));
    end
  end
  %
  %Transpose the array back to its original form
  %
  M=M';
  %
  % Calculate the number of M-ary symbols based upon the exponent qp,then use this
number
  % to find the number of equally spaced phases in a unit circle.
  N=2^qp;
  dph=2*pi/N;
  %
  % Divide the phase arguments of elements in M, by the equal phases generated by dph.
  phn=angle(M)./dph;
  %
  % Calculate the phase sector number by finding the remainders.
  %
  s=rem(round(phn)+N,N);
elseif fort==1 % Frequency Differential decoding
  %
```

```
  % Transpose the modulation array, and find the dimensions
  %
  [m,n]=size(MD);
  MD=MD(:,2:n);
  [m n]=size(MD);
  %
  % Perform a looping routine to find the phase differences between
  % adjacent values in the array, MD, and put these calculated values into array,M.
  %
  for l=1:m
    for j=1:n-1
      M(l,j)=MD(l,j+1)*conj(MD(l,j));
    end
  end
  %
  N=2^qp;
  dph=2*pi/N;
  %
  % Calculate the phase sector number by finding the remainders.
  phn=angle(M)./dph;
  s=rem(round(phn)+N,N);
end
%_____________________________________________________________________________
```

DIFCDRFT

```
%-------------------------------------------------------------------------------
%
%Title             : Complex Number Modulator & Frequency/Time Differential Encoder
%Thesis Advisor    : Prof J. McEachen, Naval Postgraduate School
%Author            : Prof. Paul H. Moose, Naval Postgraduate School
%Modified      by  : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function MD=difcdrft(q,m,fort)
if fort==0 %Time differential encoding
  %
  % M-ary alphabet size
  %
  N=2^q;
  % Determine the number of equal phases based upon the m-ary symbol length
  %
  dph=2*pi/N;
  %
  % Find the size of the input symbol matrix (# of row & # of columns)
  [rr n]=size(m);
  %
  % Perform the time differential encoding of phase values by cumulative summing
matrix,
  % m, down one column at a time across the entire matrix.  This function generates a
matrix.
  %
  for k=1:n
    md=cumsum(m(:,k));
    %
    % Generate the complex numbers with correspondiing phase values.
    %
    MD(:,k)=exp(i*dph.*md);
  end
  %
  % Inject the reference row of ones (zero phase) at top of output matrix for
  % differential encoding synchronization
  %
  MD=[ones(1,n); MD];
elseif fort==1 % Frequency Differential encoding
  %
  % M-ary alphabet size
  N=2^q;
  %
```

```
  dph=2*pi/N;
  %
  % Find the size of the input symbol matrix (# of row & # of columns)
  %
  [rr n]=size(m);
  %
  %
  md=cumsum(m');
  md=md';
  %
  % Generate the complex numbers with corresponding phase values.
  %
  MD=exp(i*dph.*md);
  %
  % Inject the reference row of ones (zero phase) at top of output matrix for
  % differential encoding synchronization.
  %
  MD=[ones(rr,2) MD];
end
%_____________________________________________________________________________
```

DIFFCHKR

```
%-------------------------------------------------------------------------------
%
%Title              : Differential Encoder/Decoder Checker
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function diffchkr(s,symno,freqno,mary,nary)
fort=input('To run the frequency version, enter 1 (one); otherwise, enter 0 (zero) to run
the time version:');
%
B=marymsg(mary,s,symno,freqno);
Rndm_m_ary_msg=B;
%
m1=bm(nary,mb(mary,B));
lengthm1=length(m1);
m=(reshape(m1,lengthm1/symno,symno))';
N_ary_msg=m;
%
if fort==1
  disp('')
  disp('Frequency Differential Encode/Decode version')
  %
  %Freq. Diff. Enc.
  %
  MDD=difcdrf(mary,m);
elseif fort~=1
  disp('')
  disp('Time Differential Encode/Decode version')
  %
  MDD=difcdrt(mary,m);
end
%
maryq=mary;
if fort==1
  %
  [s M]=difdcdrf(maryq,mary,MDD);
elseif fort ~=1
  %
  [s M]=difdcdrt(maryq,mary,MDD);
end
%
```

```
%Check results for correctness. (uses m-file check.m)
%
[error_no,freqerrs,errmx,rowerrs]=check(0,m,s,freqno,freqno,freqno);
%_____________________________________________________________________
```

DLINE

```
%-------------------------------------------------------------------------------
%
%Title                 : UHF Channel Delay Line Generator
%Thesis Advisor        : Prof J. McEachen, Naval Postgraduate School
%Author                : Prof. Paul H. Moose, Naval Postgraduate School
%Modified       by     : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function xd=dline(x,d)
x=x.';
dmax=max(d);
dmin=min(d);
nmin=floor(dmin);
nmax=ceil(dmax);
x=[x;zeros(nmax+3,1)];
N=length(x);
Nd=length(d);
%
for n=1:Nd;
   di=d(n);
   D=floor(di);
   deld=di-D;
   xd(:,n)=cvdd(x,deld-.5);
   xd(:,n)=[zeros(D,1);xd(1:N-D,n)];
end
xd=xd.';
[rr,cc]=size(xd);
xd=xd(:,4+nmin:cc);
%_______________________________________________________________________________
```

FA2CMA

```
%-------------------------------------------------------------------------------
%
%Title                : Frequency Array To Complex Modulation Array Converter
%Thesis Advisor       : Prof J. McEachen, Naval Postgraduate School
%Author               : Prof. Paul H. Moose, Naval Postgraduate School
%Modified      by     : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function Mm=fa2cma(K,X)
%
%
[m n]=size(X);
%
Mm(:,1:K)=X(:,n-K+1:n);
Mm(:,K+1:2*K)=X(:,1:K);
Cmplx_mod_vals=Mm;
%_____________________________________________________________________________
```

INTLVCHK

```
%-------------------------------------------------------------------------------
%
%Title              : Interleaver/Deinterleaver Verifier
%Thesis Advisor    : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By       : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function intlvchk(s,symno,freqno,rintlv,cintlv,mary,case)
%
multiples=mltpl(symno,freqno);
Intrlvr_nbr_mltpls=multiples;
%
%
if (symno*freqno)~=(rintlv*cintlv)
  disp('ERROR: The interleaver parameters, rintlv and cintlv, are not compatible with the
message array size.')
  disp('      The acceptable choice of possible number are:')
  disp('')
  disp(multiples)
  disp('Note: The selected pair of numbers must be divisible by the number of rows and
columns of the input matrix multiplied together.')
  disp('      In this case the number of rows times the number of columns is:')
  disp('')
  disp(symno*freqno)
elseif(symno*freqno)/(rintlv*cintlv)==1
  %
  B=marymsg(mary,s,symno,freqno);
  Random_msg=B
  %
  SYNC=[];
  [Br Bc]=size(B);
  Bt=B';
  Bvect=Bt(:)';
  si=cdlilv(rintlv,cintlv,case,Bvect,SYNC);
  Bi=reshape(si,Bc,Br)';
  Interleaved_array=Bi
  %
  [Br Bc]=size(Bi);
  SYNC=[];
  sr=Bi';
  si=sr(:)';
  sd=cdldlv(rintlv,cintlv,case,si,SYNC);
```

```
  Bd=reshape(sd,Bc,Br)';
  Deinterleaved_array=Bd
  %
  [error_no,freqerrs,errmx,rowerrs]=check(0,B,Bd,freqno,freqno,freqno);
end
%_____________________________________________________________________
```

INTLVPRS

```
%-------------------------------------------------------------------------------
%
%Title              : Intermediate Matrix Interleaver Dimension Pairs
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function pairs=intlvprs(n,m)
%
prod=n*m;
%
multvect=[1];
%
for i=2:prod;
  remdr=rem(prod,i);
  if remdr==0
    multvect=[multvect i];
  else
    multvect=multvect;
  end
  %
  mult=multvect;
end
lngth=length(mult);
nbr=mult(lngth);
result=[1 nbr];
for i=2:lngth;
  crntpr=[mult(i) nbr/mult(i)];
  result=[result;crntpr];
end
pairs=result;
%_______________________________________________________________________________
```

ITDA

```
%-------------------------------------------------------------------------------
%
%Title                 : Frequency Domain Samples Without Guard Interval
%Thesis Advisor        : Prof J. McEachen, Naval Postgraduate School
%Author                : Prof. Paul H. Moose, Naval Postgraduate School
%Modified       by     : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function Y=itda(Ng,y)
%
[L Nt]=size(y);
% Remove the guard interval for channel compensation, Ng, precursor.
%
y=y(:,Ng+1:Nt);
% Take the FFT of array,y
%
Y=fft(y.').';
%_____________________________________________
```

MARYMSG

```
%-------------------------------------------------------------------------------
%
%Title              : M-ary Message Test Pattern Generator
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function [vmary_ce,random_bit]=marymsg(q,s,n,m)
%
%for input of 100 symbols
[random_bit]=msg(s,66);
%for input of 5000 symbols
%[random_bit]=msg(s,2514);
%for input of 20000 symbols with 48 sub-carriers
%[random_bit]=msg(s,10002);
%
conv_g=[1 0 1 1 0 1 1;1 1 1 1 0 0 1];
conv_k0=1;
conv_output=cnv_encd(conv_g,conv_k0,random_bit);

vmary_ce=(reshape(bm(q,conv_output),m,n))';
%
%_______________________________________________________________________________
```

MB

```
%-------------------------------------------------------------------------------
%
%Title                 : M-ary To Binary Converter
%Thesis Advisor        : Prof J. McEachen, Naval Postgraduate School
%Author                : Prof. Paul H. Moose, Naval Postgraduate School
%Modified      by      : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function [b]=mb(q,m)
%
row=size(m,1);
col=size(m,2);
%
m=reshape(m',1,(row*col));
%
b0=rem(m,2);
m=(m-b0)./2;
B=b0;
%
%
for j=1:q-1
  bj=rem(m,2);
  m=(m-bj)./2;
  %
  %
  B=[B;bj];
end
%
b=B(:)';
binary=b;
%_______________________________________________________________________________
```

METRIC

```
%-------------------------------------------------------------------------------
%
%Title                : Viterbi Hard Decision Decoding matric
%Thesis Advisor       : Prof J. McEachen, Naval Postgraduate School
%Author               : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function distance=metric(v_x,v_y)
if v_x==v_y
  distance=0;
else
  distance=1;
end
```

MLTPL

```
%-------------------------------------------------------------------------------
%
%Title                : Common Multiples
%Thesis Advisor       : Prof J. McEachen, Naval Postgraduate School
%Author               : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function [mult]=mltpl(n,m)
%
max=n*m;
%
multvect=[1];
%
for i=2:max;
  remdr=rem(max,i);
  if remdr==0
    multvect=[multvect i];
  else
    multvect=multvect;
  end
  %
  mult=multvect;
end
%_____________________________________
```

MSG

```
%-------------------------------------------------------------------------------
%
%Title                 : Message Test Pattern Generator
%Thesis Advisor        : Prof J. McEachen, Naval Postgraduate School
%Author                : Prof. Paul H. Moose, Naval Postgraduate School
%Modified       by     : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function u=msg(s,k)
%
%rand('uniform');
%
temp=rand('seed');
%
rand('seed',s);
%
u=round(rand(1,k));
%
%_____________________________________________________________________
```

NXT_STAT

```
%--------------------------------------------------------------------------------
%
%Title              : Next State
%Thesis Advisor    : Prof J. McEachen, Naval Postgraduate School
%Author             : Tan Kok Chye, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
%
function [next_state,memory_contents]=nxt_stat(current_state,input,v_L,v_k)
binary_state=deci2bin(current_state,v_k*(v_L-1));
binary_input=deci2bin(input,v_k);
next_state_binary=[binary_input,binary_state(1:(v_L-2)*v_k)];
next_state=bin2deci(next_state_binary);
memory_contents=[binary_input,binary_state];
```

OFST

```
%-------------------------------------------------------------------------------
%
%Title                 : Channel Offset
%Thesis Advisor        : Prof J. McEachen, Naval Postgraduate School
%Author                : Prof. Paul H. Moose, Naval Postgraduate School
%Modified       by     : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function xo=ofst(e,N,x)
[m Nt]=size(x);
xo=x.';
x=x.';
x=x(:);
x=x.';
Nt=length(x);
l=1:Nt;
%
%
ex=x.*exp(i*(2*pi/N)*e.*l);
xo(:)=x;
xo=xo.';
%_____________________________________________________________________
```

RAY_DOP

```
%-------------------------------------------------------------------------------
%
%Title                  : Rayleigh Doppler
%Thesis Advisor         : Prof J. McEachen, Naval Postgraduate School
%Author                 : Prof. Paul H. Moose, Naval Postgraduate School
%Modified       by      : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function c=ray_dop(s,M,N,es)
m=0:M-1;
randn('seed',s+10);
pr1=randn(1,20);
randn('seed',s+20);
pim=i*randn(1,20);
p=pr1+pim;
p=p/(40^.5);
rand('seed',s+30);
e=rand(1,20);
e=es*cos(2*pi*(e-.5));
E=exp(i*2*pi*e'*m/N);
c=p*E;
%_____________________________________________________
```

ROTM

```
%-------------------------------------------------------------------------------
%
%Title                : Rotate Vector
%Thesis Advisor       : Prof J. McEachen, Naval Postgraduate School
%Author               : Prof. Paul H. Moose, Naval Postgraduate School
%Modified      by     : Tan Kok Chye, Naval Postgraduate School
%
%-------------------------------------------------------------------------------
%
function [vp,vn]=rotm(v,m)
L=length(v);
m=rem(m,L);
ii=(1:L)-1;
isp=rem(ii-m+L,L)+1;
isn=rem(ii+m+L,L)+1;
vp=v(isp);
vn=v(isn);
%_____________________________________________________________
```

TDA

```
%--------------------------------------------------------------------------------
%
%Title                : Time Domain Samples With Guard Interval Precursor
%Thesis Advisor       : Prof J. McEachen, Naval Postgraduate School
%Author               : Prof. Paul H. Moose, Naval Postgraduate School
%Modified      by     : Tan Kok Chye, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
%
function x=tda(Ng,X)
[m N]=size(X);
%
% Perform inverse FFT on frequency values in array,X
%
x=ifft(X.');
% Add precursor of Ng samples to the beginning of the time domain array for channel
% compensation.
%
x=x.';
if Ng==0
   x=x;
else
   x=[x(:,N-Ng+1:N)x];
end
%_____________________________________________________________________________
```

UHFIFT

```
%--------------------------------------------------------------------------------
%
%Title              : Channel 2 Simulation w/o Interleaving
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
%
function
[errors,freqerrs]=uhfift(picy_n,pic,s,freqno,rintlv,cintlv,N,mary,nary,n,k,blklgth,Ng,loss,
dly,dop,freqspace,fort)
%
[xmt,modvals,B,nsymno]=coderift(picy_n,pic,s,freqno,rintlv,cintlv,N,mary,nary,fort);
%
xmtifft=tda(Ng,xmt);
xmtpts=1:length(xmtifft);
%
if picy_n==1
   xmtpts=1:length(xmtifft);
   figure(3)
   plot(xmtpts,xmtifft)
   title('Transmitted Time Domain Signal')
   axis('square');
   orient tall
   grid
end
%
%
sandn=chuhf(s+1,xmtifft,loss,dly,dop,N,freqspace);
%
%
if picy_n==1
   rcvdpts=1:length(sandn);
   figure(4)
   plot(rcvdpts,sandn)
   title('Received Time Domain Signal')
   axis('square');
   orient tall
   grid
end
%
%
sandnfft=itda(Ng,sandn);
```

```
%
%
K=(length(modvals(1,:)))/2;
rdintlv=rintlv;
cdintlv=cintlv;
rcvd=decdrift(picy_n,pic,K,sandnfft,nsymno,freqno,rdintlv,cdintlv,mary,nary,fort);
Transmitted_msg=B;
Received_msg=rcvd;
%
%
[errors,freqerrs,errmx,rowerrs]=check(pic,B,rcvd,n,k,blklgth);
errmx;
[rn cm]=size(errmx);
errsum=sum(errors);
if errsum~=0
  symno=rintlv*cintlv/freqno;
  freqno;
  if picy_n==1
    if dop==[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15]
      figure(2)
      mesh(errmx)
      title(['Link 3: Error Distribution Without Interleaving (M-ary bits:
',int2str(mary),',','N-ary bits:',int2str(nary),')'])
      axis([0 freqno 0 symno 0 max(max(errmx))])
      xlabel(['Freq.# (Total=',int2str(freqno),')'])
      ylabel(['Row#(Symbol#=',int2str(symno*freqno),')'])
      zlabel(['Error Occurance (Total =',int2str(errsum),')(seed =',num2str(s),')'])
      text(-150,0,1.95,['Error Correction =',int2str(floor((n-k)/2))])
      grid
      orient tall
      %
      %
    elseif dop==[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10]
      figure(2)
      mesh(errmx)
      title(['Link 2: Error Distribution Without Interleaving (M-ary
bits:',int2str(mary),',','N-ary bits:',int2str(nary),')'])
      axis([0 freqno 0 symno 0 max(max(errmx))])
      xlabel(['Freq.# (Total=',int2str(freqno),')'])
      ylabel(['Row#(Symbol#=',int2str(symno*freqno),')'])
      zlabel(['Error Occurance (Total =',int2str(errsum),')(seed =',num2str(s),')'])
      text(-150,0,1.95,['Error Correction =',int2str(floor((n-k)/2))])
      grid
      orient tall
      %
      %
```

```
        elseif dop==[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5]
        figure(2)
        mesh(errmx)
        title(['Link 1: Error Distribution Without Interleaving (M-ary
bits:',int2str(mary),',','N-ary bits:',int2str(nary),')'])
        axis([0 freqno 0 symno 0 max(max(errmx))])
        xlabel(['Freq.# (Total=',int2str(freqno),')'])
        ylabel(['Row#(Symbol#=',int2str(symno*freqno),')'])
        zlabel(['Error Occurance (Total =',int2str(errsum),')(seed =',num2str(s),')'])
        text(-150,0,1.95,['Error Correction =',int2str(floor((n-k)/2))])
        grid
        orient tall
        %
        %
      end
    end
  else
    disp('')
    disp('GREAT!!! Test passed.')
  end
  if sum(rowerrs)~=0
    figure(3)
    cony=(max(rowerrs)+5)/60;
    conx=symno/80;
    errindx=1:length(rowerrs);
    bar(errindx,rowerrs)
    title(['Error Count Per Symbol Row (Total Errors=',int2str(sum(rowerrs)),')'])
    xlabel('Row Number')
    ylabel('Number of Errors')
    axis([0.5,(symno+.5),0,(max(rowerrs)+(6*cony))])
    if sum(rowerrs)~=0
      for i=1:length(rowerrs)
        text(i-(1.5*conx),rowerrs(i)+(4*cony),int2str(rowerrs(i)))
      end
    end
    orient tall
  end
  %_____________________________________________________________________________
```

UHFSEEDS

```
%--------------------------------------------------------------------------------
%
%Title              : Seed Error Report
%Thesis Advisor     : Prof J. McEachen, Naval Postgraduate School
%Author             : Dave Roderick, Naval Postgraduate School
%Modified By        : Tan Kok Chye, Naval Postgraduate School
%
%--------------------------------------------------------------------------------
disp('_____________________________________________________________________________
___');
fort=input('To run the frequency version, enter 1 (one); otherwise, enter 0 (zero) to run
the time version:');
pthno=input('Do you want to run link1, link2, link3 or a custom link? (Enter 1,2,3 or 4
for custom):');
%
%
if pthno==3
  %my link 3

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
    dop=[15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

elseif pthno==2
  %my link 2

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
    dop=[10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10];

dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

elseif pthno==1
  %my link 1

loss=[0,2.17,4.34,6.51,8.69,10.86,13.03,15.20,17.37,19.54,21.71,23.89,26.06,28.23,30.4,
32.57,34.74,36.92];
    dop=[5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5];
```

```
dly=[0,0.05,0.10,0.15,0.20,0.25,0.30,0.35,0.40,0.45,0.50,0.55,0.60,0.65,0.70,0.75,0.80,0.
85];

elseif pthno==4
   disp('Custom link simulation..')
   loss=input('Enter the path loss in dB (Ex. [0 4 7]):');
   dop=input('Enter the doppler frequency in Hertz (Ex. [30 20 15]):');
   dly=input('Enter the time delays of the multipaths in microsecs (Ex. [0 0.6 3.9]):');
end
prnty_n=input('Do you want print outs? (1=yes, 0=no):');
%
%
symbols=input('Enter the minimum number of symbols to test:');
freqno=input('Enter the number of FFT points (NOTE: Must be larger than # of OFDM
frequencies):');
N=input('Enter the number of FFT points (Note : Must be larger than # of OFDM
frequencies):');
smax=input('All tested seeds begin with one and end with a max number. Enter Smax
(Integer#):');
disp(['Tested seed range is 1 - ',int2str(floor(smax)),'...'])
mary=8;
nary=4;
symno=ceil(symbols/freqno);
freqspc=16600000/freqno;
errvect=[];
incvect=[];
topervect=[];
sindex=1:floor(smax);
for s=sindex;
   %
   %

[errors,freqerrs]=uhfift(0,0,s,freqno,freqno,symno,N,mary,nary,0,0,freqno,6,loss,dly,dop,
freqspc,fort);
   errtot=sum(errors);
   errvect=[errvect,errtot];
end
totalerr=sum(errvect);
avgerr=ceil(totalerr/floor(smax));
[inc I]=sort(errvect);
errmx=[I;inc]
Error_Seeds=incvect
Total_Errors=totalerr
Avg_Errors=avgerr
save unfhist errmx
```

```
disp('All Done!!!')
disp('')
if sum(inc)==0
   disp('GREAT!!! Simulation passed with no errors.')
elseif sum(inc)~=0
   disp('OOOPS!!! Errors occured.')
end
%
% Plot
%
figure(3)
bar(sindex,errvect)
grid
orient tall
xlabel(['UHFSEEDS: Seed Value (Symbol#=',int2str(freqno*symno),')'])
ylabel(['Error Number (OFDM Freq.#=',int2str(freqno),')(M-ary=',int2str(2^mary),',N-
ary=',int2str(2^nary),')'])
if fort==1
   if pthno==1
      title(['Link1:Error Dist. vs. Seed Values (Freq. Diff. Enc.)
(Loss=',num2str(loss),')(Dop=',num2str(dop),')(Delay=',num2str(dly),')']);
   elseif pthno==2
      title(['Link2:Error Dist. vs. Seed Values (Freq. Diff. Enc.)
(Loss=',num2str(loss),')(Dop=',num2str(dop),')(Delay=',num2str(dly),')']);
   elseif pthno==3
      title(['Link3:Error Dist. vs. Seed Values (Freq. Diff. Enc.)
(Loss=',num2str(loss),')(Dop=',num2str(dop),')(Delay=',num2str(dly),')']);
   elseif pthno==4
      title(['Custom Link:Error Dist. vs. Seed Values (Freq. Diff. Enc.)
(Loss=',num2str(loss),')(Dop=',num2str(dop),')(Delay=',num2str(dly),')']);
   end
elseif fort~=1
   if pthno==1
      title(['Link1:Error Dist. vs. Seed Values (Freq. Diff. Enc.)
(Loss=',num2str(loss),')(Dop=',num2str(dop),')(Delay=',num2str(dly),')']);
   elseif pthno==2
      title(['Link2:Error Dist. vs. Seed Values (Freq. Diff. Enc.)
(Loss=',num2str(loss),')(Dop=',num2str(dop),')(Delay=',num2str(dly),')']);
   elseif pthno==3
      title(['Link3:Error Dist. vs. Seed Values (Freq. Diff. Enc.)
(Loss=',num2str(loss),')(Dop=',num2str(dop),')(Delay=',num2str(dly),')']);
   elseif pthno==4
      title(['Custom Link:Error Dist. vs. Seed Values (Freq. Diff. Enc.)
(Loss=',num2str(loss),')(Dop=',num2str(dop),')(Delay=',num2str(dly),')']);
   end
end
```

```
axis([.5 (max(s)+.5) 0 (max(errvect)+1)])
if prnty_n==1
   print
   pause(10)
end
figure(4)
bar(inc)
grid
orient tall
xlabel('UHFSEEDS: Seed Value (out of order)')
ylabel('Error Number')
title('Ordered Error Dist. vs. Seed Values (Corresponding Seed Shown on Plot)')
axis([.5 (max(s)+.5) 0 (max(errvect)+1)])
for i=1:length(errvect)
   if inc(i)>(max(inc))*.8
      incvect=[incvect,I(i)];
      topervect=[topervect,inc(i)];
      errlth=length(topervect);
      yinc=(max(inc(i))-min(inc(i)))/2;
      text(5,(inc(i)+1),int2str(I(i)))
   end
end
if prnty_n==1
   print
   pause(10)
end
figure(5)
hist(errvect)
title(['Error Histogram (Average# of Errors Per Seed=',int2str(avgerr),')'])
xlabel('Error Bins')
ylabel('Number of Seeds')
grid
orient tall
if prnty_n==1
   print
end
%_____
```

VITERBI

```
%-------------------------------------------------------------------------------
%
%Title                 : Viterbi Decoder
%Thesis Advisor    : Prof J. McEachen, Naval Postgraduate School
%Reference          : Contemporary Communication System using MatLab
%John G. Proakis & Masoud Salehi.
%-------------------------------------------------------------------------------
%
function
[decoder_output,survivor_state,cumulated_metric]=viterbi(v_G,v_k,channel_output)
%
v_n=size(v_G,1);
% check the sizes
if rem(size(v_G,2),v_k)~=0
  error('Size of v_G and v_k do not agree')
end
if rem(size(channel_output,2),v_n)~=0
  error('channel output not of the right size')
end
v_L=size(v_G,2)/v_k;
number_of_states=2^((v_L-1)*v_k);
%generate state transition matrix, output matrix, and input matrix
for v_j=0:number_of_states-1
  for v_l=0:2^v_k-1
    [next_state,memory_contents]=nxt_stat(v_j,v_l,v_L,v_k);
    input(v_j+1,next_state+1)=v_l;
    branch_output=rem(memory_contents*v_G',2);
    nextstate(v_j+1,v_l+1)=next_state;
    output(v_j+1,v_l+1)=bin2deci(branch_output);
  end
end
state_metric=zeros(number_of_states,2);
depth_of_trellis=length(channel_output)/v_n;
channel_output_matrix=reshape(channel_output,v_n,depth_of_trellis);
survivor_state=zeros(number_of_states,depth_of_trellis+1);
%start decoding of non-tail channel outputs
for v_i=1:depth_of_trellis-v_L+1
  flag=zeros(1,number_of_states);
  if v_i<=v_L
    step=2^((v_L-v_i)*v_k);
  else
    step=1;
  end
  for v_j=0:step:number_of_states-1
```

```
    for v_l=0:2^v_k-1
      branch_metric=0;
      binary_output=deci2bin(output(v_j+1,v_l+1),v_n);
      for v_ll=1:v_n

branch_metric=branch_metric+metric(channel_output_matrix(v_ll,v_i),binary_output(v_
ll));
      end

if((state_metric(nextstate(v_j+1,v_l+1)+1,2)>state_metric(v_j+1,1)+branch_metric)|flag(
nextstate(v_j+1,v_l+1)+1)==0)
        state_metric(nextstate(v_j+1,v_l+1)+1,2)=state_metric(v_j+1,1)+branch_metric;
        survivor_state(nextstate(v_j+1,v_l+1)+1,v_i+1)=v_j;
        flag(nextstate(v_j+1,v_l+1)+1)=1;
      end
    end
  end
  state_metric=state_metric(:,2:-1:1);
end
%start decoding of the tail channel_outputs
for v_i=depth_of_trellis-v_L+2:depth_of_trellis
  flag=zeros(1,number_of_states);
  last_stop=number_of_states/(2^((v_i-depth_of_trellis+v_L-2)*v_k));
  for v_j=0:last_stop-1
    branch_metric=0;
    binary_output=deci2bin(output(v_j+1,1),v_n);
    for v_ll=1:v_n

branch_metric=branch_metric+metric(channel_output_matrix(v_ll,v_i),binary_output(v_
ll));
    end
    if
((state_metric(nextstate(v_j+1,1)+1,2)>state_metric(v_j+1,1)+branch_metric)|flag(nextst
ate(v_j+1,1)+1)==0)
      state_metric(nextstate(v_j+1,1)+1,2)=state_metric(v_j+1,1)+branch_metric;
      survivor_state(nextstate(v_j+1,1)+1,v_i+1)=v_j;
      flag(nextstate(v_j+1,1)+1)=1;
    end
  end
  state_metric=state_metric(:,2:-1:1);
end
%generate the decoder output from the optimal path
state_sequence=zeros(1,depth_of_trellis+1);
state_sequence(1,depth_of_trellis)=survivor_state(1,depth_of_trellis+1);
for v_i=1:depth_of_trellis
```

```
   state_sequence(1,depth_of_trellis-
v_i+1)=survivor_state((state_sequence(1,depth_of_trellis+2-v_i)+1),depth_of_trellis-
v_i+2);
end
decoder_output_matrix=zeros(v_k,depth_of_trellis-v_L+1);
for v_i=1:depth_of_trellis-v_L+1
   dec_output_deci=input(state_sequence(1,v_i)+1,state_sequence(1,v_i+1)+1);
   dec_output_bin=deci2bin(dec_output_deci,v_k);
   decoder_output_matrix(:,v_i)=dec_output_bin(v_k:-1:1)';
end
decoder_output=reshape(decoder_output_matrix,1,v_k*(depth_of_trellis-v_L+1));
cumulated_metric=state_metric(1,1);
%-------------------------------------------------------------------------------------------------------------
```

LIST OF REFERENCES

1. Yun, Xiaoping & Lewis, *Ted, Feasibility Analysis of Deploying Wireless LAN Onboard Submarines and Surface Ships*, Naval Postgraduate School Research, Volume 9, No. 3, pg.1, Monterey, CA, October 1999.

2. Bob O'Hara & Al Petrick, *IEEE 802.11 Handbook: A Designer's Companion*, Standards Information Network IEEE Press, 3 Park Avenue, NY, 1999.

3. R Van Nee & R. Prasad, *OFDM for Mobile Multimedia Communications*, Artech House, MA, 1999

4. T.S. Rappaport, S.Y. Seidel & K. Takamizawa, "Statistical Channel Impulse Response Models for Factory and Open Plan Building Radio Communication System Design," *IEEE Transactions on Communications*, vol.39, pp. 794-807, May 1991.

5. Bernard Sklar, *Digital Communications Fundamentals and Applications*, Prentice Hall, Upper Saddle River, NJ, 2000.

6. P. Shelswell, M.A, *The COFDM Modulation System, The Heart of Digital Audio Broadcasting*, BBC Research and Development Report, 1995

7. William Y. Zou & Yiyan Wu, "COFDM An Overview," *IEEE Transactions on Broadcasting*, Vol. 41, No. 1, pp. 1-8, March 1995.

8. Leonard J. Cimini, "Analysis and Simulation of a Digital Mobile Channel Using Othogonal Frequency Division Multiplexing," *IEEE Transactions on Broadcasting*, Vol. 33, No. 1, pp. 665-675, July 1985.

9. E. Oran Brigham, *The Fast Fourier Transform*, Prentice Hall, Inc, Englewood Cliffs, NJ, 1974

10. Masato, Kiyoshi & Takeshi, "The Performance of a Packet Mode OFDM Modem for 5 GHz Band High-Data-Rate Wireless LANs," *IEEE Transactions on Broadcasting*, Vol 24, No. 1, pp. 295-299, June 1999.

11. Theodore S. Rappaport, *Wireless Communications*, Prentice Hall, Upper Saddle River, NJ, 2000.

12. Ramjee Prasad, *CDMA For Wireless Personnel Communications*, Artech House, MA, 1996.

13. C.W. Farrow, "A Continuously Variable Digital Delay Element," *Proceedings of IEEE International Symposium on Circuits and Systems*, NY, pp. 2641-2645, May 1988.

14. Louis Thibault & Minh Thien Le, "Performance Evaluation Of COFDM For Digital Audio Broadcasting Part I : Parametric Study," *IEEE Transactions on Broadcasting*, Vol 43, No. 1, pp. 64-75, March 1997.

15. R. Hoshyar, S. H. Jamali & A. R. S. Bahai, "Turbo Coding Performance In OFDM Packet Transmission," *IEEE Transactions on Broadcasting*, Vol 21, No. 1, pp.805-810, July 2000.

16. Ramakrishna Janaswamy, *Radiowave Propagation And Smart Antennas For Wireless Communications*, Kluwer Academic Publishers, Norwell, MA, 2000.

17. David V. Roderick, *A Coded Orthogonal Frequency Division Multipath Simulation Of A High Data Rate, Line-Of-Sight, Digital Radio For Mobile Maritime Communications*, Master's Thesis, Naval Postgraduate School, Monterey, CA, June 1997.

18. Benny Bing, *Wireless Local Area Networks*, Artech House, MA, 2000.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
   8725 John J. Kingman Road, Suite 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library.....2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Chairman, Code EC……...……..…..…….1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. John McEachen, Code EC/MJ.. . . . . .…... 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

5. Xiaoping Yun, Code EC/YX. .. . .…..…. 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

6. Engineering and Technology Curriculum, Code 34…....……..……..1
   Naval Postgraduate School
   700 Dyer Road, Room 115
   Monterey, CA 93943-5107

7. Ms Winnifer ... .…..…. . . . . .……..2
   Defence Science & Technology Agency
   71 Science Park Drive
   #02-05, Singapore 118253

8. Voo Lin Mei. . . . . .…..…. . . . . .……...2
   BLK 725, Ang Mo Kio, Ave 6, #12-4142
   Singapore (560725)

9. Toh Ban Huat. . . . . .…..…. . . . . .…….1
   BLK 117, Serangoon North, Ave 1, #09-241
   Singapore (550117)

10. Department Of Defense .. . .…..…. . . . . .…….1
9800 Savage Rd
Ft. Meade, MD 20755
ATTN : R531

11. Naval Engineering Logistics Office……..1
4555 Overlook Ave SW
Code 5707
Washington DC 20375-5707